# Resource Scheduling Strategies to Optimise QoS in Integrated IoT and Fog Computing Environments

**Naif Alshammari**
School of Computing and Communications
Lancaster University

A thesis submitted for the degree of
*Doctor of Philosophy*

August, 2025

I dedicate this thesis to the souls of my mother and father.

# Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. This thesis does not exceed the maximum permitted word length of 80,000 words, including appendices and footnotes, but excluding the bibliography.

Naif Alshammari

# Abstract

In recent years, there has been a tremendous increase in the internet and its applications, which has attracted the attention of scholars and industries to investigate how to improve the quality of services (QoS). Improving QoS is considered a major challenge for users of IoT devices. To address this challenge, several technologies have emerged to extend cloud computing, including fog computing, which provides computational resources at the network's edge. Nevertheless, fog computing faces limitations due to the constrained resources, limited computational processes and storage in fog devices compared to cloud infrastructure.

This thesis investigates resource scheduling strategies to optimise QoS in fog computing, focusing on task scheduling and resource allocation approaches. The thesis begins with a qualitative comparative analysis of existing resource management approaches to optimise QOS. It classifies resource management approaches into several categories: application placement, task scheduling, resource allocation, task offloading, load balancing, and resource provisioning. These categories are either task-oriented, such as application placement, task scheduling, and task offloading, or resource-oriented, including resource allocation, load balancing, and resource provisioning.

It also introduces a novel intelligent resource scheduling model using gated graph convolution neural networks (GGCNs) to trade off between delay and network usage with a limited number of fog nodes. The GGCN model outperforms various other existing approaches like PSO, FCFS, and JSF by 86.09%, 98.53%, and 98.02% respectively, in terms of total network usage. Additionally, in terms of loop delay, it achieves improvements of 68.64% over PSO, 92.07% over FCFS, and 76.26% over SJF.

Furthermore, it presents a novel multi-objective scheduling framework utilising an enhanced multi-layer perceptron (eMLP). This new mechanism optimises several parameters, including delay, power consumption, and cost, while simultaneously optimising bandwidth. Experimental results show that eMLP reduces delay, network usage and cost by 75%, 65%, and %70 respectively, compared to other benchmark schemes such as GNN, SMA, FCFS, and SJF.

Finally, the thesis discusses the current gaps and future directions for enhancing and further investigating QoS through fog computing.

# Publications

Naif Alshammari, Haris Pervaiz, Hasan Ahmed, and Qiang Ni, "Delay and Total Network Usage Optimisation Using GGCN in Fog Computing." in *IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2023.

Naif Alshammari, Sukhpal Gill, Haris Pervaiz, Qiang Ni, Hasan Ahmed, "Resource Scheduling in Integrated IoT and Fog Computing Environments: A Taxonomy, Survey and Future Directions." In: Mukherjee, A., De, D., Buyya, R. (eds) *Resource Management in Distributed Systems. Studies in Big Data, vol 151. Springer,* Singapore, 2024.

Mohil Patel, Sudeep Tanwar, Anish Jindal, Naif Alshammari, Haris Pervaiz, Hasan Ahmed, "A Blockchain-based Predictive Maintenance Scheme for Smart Agriculture" in *17th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, Sharjah, UAE, 2024 (Accepted/in Press).

Naif Alshammari, Haris Pervaiz, Hasan Ahmed, and Qiang Ni, "Enhancing QoS in Fog Computing via Enhanced Multilayer Perceptron-Based Multi-Objective Task Scheduling." Submitted to *IEEE Internet of Things Journal* .

# Acknowledgements

I would like to sincerely thank my friends and officemates at Infolab for their constant support and encouragement throughout my PhD journey. I would also like to extend my gratitude to Dr. Haris Pervaiz, Dr. Hasan Ahmed, and Professor Qiang Ni for their support and supervision, especially Dr.Haris, who continued supervising me even after moving from Lancaster University. Additionally, I thank Dr. Yehia Elkhatib for his support, valuable tips and genuine care. Special appreciation goes to my family—especially my wife, children, and my brothers and sisters—for their endless patience, love and inspiration.

# Contents

**declaration** **ii**

**abstract** **iii**

**publication** **iv**

**acknowledgements** **v**

**List of Figures** **x**

**List of Table** **xi**

**List of Acronyms** **xvi**

**List of Mathematical Operators and symbols** **xviii**

**1 Introduction** **1**
1.1 Background . . . . . 1
1.2 Cloud Computing: Overview and Challenges . . . . . 2
1.3 Edge Computing: Overview and Challenges . . . . . 3
1.4 Fog Computing: Overview and Architecture . . . . . 5
1.4.1 Architecture and Key Concepts . . . . . 6
1.4.2 Comparative Analysis of Cloud, Edge and Fog . . . . . 7
1.5 Motivation . . . . . 9
1.6 Research Questions . . . . . 10
1.7 Contributions and Methodology . . . . . 11
1.8 Thesis Outline . . . . . 12

**2 Task-Oriented Approaches** **13**
2.1 Motivation . . . . . 13
2.2 Resource Management in Fog Computing . . . . . 13
2.2.1 Application Placement . . . . . 14

2.2.2 Task Scheduling . . . 26
2.2.3 Task Offloading . . . 32
2.3 Conclusion . . . 40

**3 Advanced Resource Management 41**
3.1 Resource-Oriented Management Approaches . . . 41
3.1.1 Resource Allocation . . . 41
3.1.2 Load Balancing . . . 53
3.1.3 Resource Provisioning . . . 58
3.2 Simulations in Fog Computing: Advantages . . . 65
3.2.1 Overview of Key Simulation Tools in Fog Computing . . . 66
3.3 Conclusion . . . 72

**4 Enhancing QoS using GGCN-Based Resource Allocation in Fog Computing Environment 75**
4.1 Motivation . . . 75
4.1.1 Key Contributions . . . 76
4.1.2 Overview of GGCN . . . 76
4.2 Resource Scheduling Strategies . . . 77
4.2.1 Shortest Job First . . . 77
4.2.2 First Come First Served . . . 78
4.2.3 Particle Swarm Optimization . . . 79
4.3 Proposed GGCN Methodology . . . 81
4.3.1 Problem Formulation . . . 82
4.3.2 Proposed GGCN based Resource Scheduler . . . 84
4.4 Performance Evaluation . . . 86
4.4.1 Experiment Setup . . . 87
4.4.2 Configuration . . . 87
4.4.3 Case studies . . . 88
4.4.4 Results and Discussion . . . 90
4.5 Challenges and Limitations . . . 94
4.6 Conclusion . . . 95

**5 eMLP-Based Task Scheduler to Optimize QoS in Fog Computing Environment 96**
5.1 Motivation . . . 96
5.2 Overview . . . 96
5.3 Resource Scheduling Strategies . . . 97
5.3.1 Shortest Job First . . . 97
5.3.2 First Come First Served . . . 98
5.3.3 Graph Neural Network . . . 98

5.3.4 Stable Matching Algorithm . . . . . 100
5.3.5 MLP: Versatile Application . . . . . 101
5.4 Problem Formulation . . . . . 101
5.5 System Model . . . . . 105
5.5.1 eMLP Scheduler . . . . . 107
5.6 Performance Evaluation . . . . . 110
5.6.1 Experimental Setup . . . . . 110
5.6.2 Configuration . . . . . 111
5.6.3 Results and Discussion . . . . . 112
5.7 Conclusion and Future Work . . . . . 117

**6 Conclusion and Future Directions** **118**

**References** **120**

# List of Figures


1.1 Challenges of Cloud Computing . . . . . 3
1.2 Advantages of Fog Computing for Resource Scheduling Strategies . . 5
1.3 Fog Computing Architecture . . . . . 6
1.4 Hierarchical architecture illustrating the spatial distribution of data processing across Edge Devices, Edge Computing, Fog Computing, and Cloud computing layers . . . . . 9

2.1 Taxonomy of Application Placement Approaches . . . . . 16
2.2 Hierarchical Representation of Task Scheduling Approaches . . . . . 27
2.3 Task Offloading Types . . . . . 33

3.1 Hierarchical Representation of Resource Allocation Mechanisms . . . 42
3.2 Strategies to Resolve Load Balancing . . . . . 55
3.3 Types of Resource Provisioning . . . . . 60

4.1 System Architecture . . . . . 81
4.2 GGCN Architecture . . . . . 84
4.3 Fog Computing Topology Designed in iFogSim . . . . . 87
4.4 Performance Analysis of Proposed GGCN: Total Network Usage and Average Loop Delay Compared Benchmark Schemes (Case study A) 91
4.5 Performance Analysis of Proposed GGCN: Total Network Usage and Average Loop Delay Compared Benchmark Schemes (Case study B) 91
4.6 Performance Analysis of Proposed GGCN: Total Network Usage and Average Loop Delay Compared Benchmark Schemes (Case study C) 92
4.7 Performance Analysis of Proposed GGCN: Total Network Usage and Average Loop Delay Compared Benchmark Schemes (Case study D) 92

5.1 System Model . . . . . 106
5.2 Systematic Block Diagram of the Proposed Enhanced Multilayer Perceptron (eMLP) . . . . . 109

5.3 Performance Analysis of the Proposed eMLP Average Delay Compared to Multiple Benchmark Schemes Across Different Task Loads . . . . . 113
5.4 Performance Analysis of the Proposed eMLP Network Usage Compared to Multiple Benchmark Schemes Across Different Task Loads . . . . . 114
5.5 Performance Analysis of the Proposed eMLP Cost of Usage Compared to Multiple Benchmark Schemes Across Different Task Loads . . . . . 115
5.6 Performance Analysis of the Proposed eMLP Average Power Consumption Compared to Multiple Benchmark Schemes Across Different Task Loads . . . . . 116
5.7 Percentage Improvement of the Proposed eMLP Over Benchmark schemes(FCFS, SJF, GNN, SMA) . . . . . 117

# List of Tables


1.1 Comparison of Cloud, Fog, and Edge Computing:Advantages and Challenges [19, 10, 13] . . . . . 8

2.1 Overview of Case Studies and Proposed Mechanism in Application Placement Approaches (Part 1) . . . . . 23
2.2 Overview of Case Studies and Proposed Mechanism in Task Scheduling Approaches (Part1) . . . . . 30
2.3 Overview of Case Studies and Proposed Mechanism in Task Offloading Approaches (Part 1) . . . . . 37

3.1 Overview of Case Studies and Proposed Mechanism in Resource Allocation Approaches (Part 1) . . . . . 49
3.2 Overview of Case Studies and Proposed Mechanism in Load Balancing Approaches (part 1) . . . . . 57
3.3 Overview of Case Studies and Proposed Mechanism in Resource Provisioning Approaches (Part 1) . . . . . 63
3.4 Common Simulators for Fog Computing Environments . . . . . 72

4.1 Simulation Setup . . . . . 87
4.2 Configurations . . . . . 88

5.1 Simulation Setup . . . . . 110
5.2 Characteristics of Fog Nodes . . . . . 112

# List of Acronyms

| | |
|---|---|
| **A2C** | Advanced Actor-Critic. |
| **A3C** | Asynchronous Advantage Actor-Critic. |
| **ABC-JAVA** | A hybrid algorithm combining ABC and JAVA algorithms. |
| **ABC** | Artificial Bee Colony. |
| **ACA** | Ant Colony Algorithm. |
| **ACO** | Ant Colony Optimization. |
| **AI** | Artificial Intelligence. |
| **ANFIS** | Adaptive Neuro-Fuzzy Inference System. |
| **ARU** | Average Resource Utilisation. |
| **AWRR** | Adaptive Weighted Round Robin. |
| **B&B** | Branch and Bound. |
| **BASP** | Bandwidth and Availability-Aware Service Placement. |
| **Bayes' Classifier** | A probabilistic classifier that uses Bayes' Theorem to predict the category of new task based on historical data. |
| **BFD** | The Best Fit Decreasing Algorithm. |
| **BF** | Best Fit. |
| **BLA** | Bee Life Algorithm. |
| **BPSO-FMP** | Binary Particle Swarm Optimization for Fog Module Placement. |
| **CB-E** | Communication-Based-Efficient. |
| **CDDQL** | Clipped Double Deep Q-learning. |
| **CGOA** | Chaotic-Based Grasshopper Optimization Algorithm. |
| **CML** | Collaborative Machine Learning. |
| **CNN** | Convolution Neural Network. |
| **CNs** | Community Networks. |
| **COA** | Crayfish Optimization Algorithm. |
| **Convex Optimization** | A subclass of optimization problems where the objective function is convex, making them easier to solve. |
| $CO_2$ | Carbon Dioxide. |
| **CPU** | Central Processing Units. |
| **CS** | Cuckoo Search Algorithm. |
| **CSA** | Crow Search Algorithm. |
| **CSP** | Cloud Service Provider. |
| **CTFS** | Critical Task First. |
| **D2D** | Device-to-Device. |
| **DAGs** | Directed Acyclic Graphs. |
| **DALBFog** | Deadline-Aware Load-Balancing Fog Computing Algorithm. |

| | |
|---|---|
| **DCTO** | Dynamic Collaborative Task Offloading. |
| **DDQL** | Double Deep Q-Learning. |
| **DEBTS** | Delay Energy Balanced Task Scheduling Algorithm. |
| **DECM** | Dynamic Energy-Efficient Cloudlet Management. |
| **DFTLA** | Dynamic Fault-Tolerant Learning Automata. |
| **DL** | Deep Learning. |
| **DLA-FMP** | Distributed Learning Automata for Fog Module Placement. |
| **DLQBRA** | Deep Q-network based Resource Allocation. |
| **DP-I** | Delay Priority-Independent. |
| **DQN** | Deep Q-Learning. |
| **DRAM** | Dynamic Resource Allocation. |
| **DRL** | Deep Reinforcement Learning. |
| **DRM** | Data Replica Manager. |
| **Dynafog** | Dynamic Task Offloading Framework for IoT-based Fog Computing Platforms. |
| **e-OPEX** | Operational Expenditure. |
| **E** | Edge. |
| **EETSPSO** | Energy Efficient Task Scheduling in Fog Computing Based on Particle Swarm Optimization. |
| **EFRO** | Energy-Aware Fog Resource Optimization. |
| **EGA-FMP** | Elitism-Based Genetic Algorithm for Fog Module Placement. |
| **EGA** | Enhanced Genetic Algorithm. |
| **EGAPSO** | Elitism-Based Genetic Algorithm and Particle Swarm Optimization. |
| **ELBS** | Energy-Aware Load Balancing and Scheduling. |
| **eMLP** | Enhanced Multilayer Perceptron. |
| **EMOPSOC** | Enhanced multi-objective Particle Swarm Optimization with Clustering. |
| **EMS** | Enterprise Management Systems. |
| **ENORM** | Edge Node Resource Management. |
| **EPRAM** | Effective Prediction and Resource Allocation Methodology. |
| **EPSO** | Extended Particle Swarm Optimization. |
| **ERAS** | Effective Resource Allocation Strategy. |
| **F-RAN** | Fog Radio Access Network. |
| **FA** | Firefly Algorithm. |
| **FCFS** | First Come First Served. |
| **FF** | First Fit. |
| **FFD** | First Fit Decreasing. |
| **FFRPP** | Fog-Aware File Replica Placement Policy. |
| **FGN** | Fog Gateway Node. |

| | |
|---|---|
| **FIFO** | First in First out. |
| **Fission-Fusion** | A social structure where the size and composition of a group change dynamically over time-based on the tasks being performed. |
| **FMFK** | Federated Multidimensional Fractional Knapsack. |
| **Fog-MMKP** | Fog Computing with Multidimensional Multiple Knapsack Problem. |
| **FONS** | Fog Orchestrator Node Selection. |
| **FRAS** | Fuzzy-Based Real-Time Auto-Scaling. |
| **Fronthaul** | The connection between distributed radio units and the centralized processing units in a network. |
| **FTLBSA** | Fault-Tolerant Load-Balancing Scheduling Algorithm. |
| **FTO** | Fair Task Offloading. |
| **FWA** | Fireworks Algorithm. |
| **GA** | Genetic Algorithm. |
| **GAAPSO-FMP** | Genetic Algorithm and Particle Swarm Optimization for Fog Module Placement. |
| **GABVMP** | Genetic Algorithm-Based Virtual Machine Placement. |
| **GGCNs** | Gated Graph Convolutional Neural Networks. |
| **GNN** | Graph Neural Network. |
| **Guifi.net** | A large community network located in Catalonia, Spain. |
| **HABBP** | Hungarian Algorithm-Based Binding Policy. |
| **HEFT** | Heterogeneous Earliest Finish Time. |
| **HHRA** | Hyper-Heuristic Resource Allocation. |
| **HLS** | High-Level Strategy. |
| **HMM** | Hidden Markov Model. |
| **HPSOFF-RPT** | Hybrid Particle Swarm Optimization with Firefly-based Resource Provisioning Technique. |
| **IC** | Incentive Compatibility. |
| **IIoT** | Industrial Internet of Things |
| **ILP** | Integer Linear Programming. |
| **IMPALA** | Importance-Weighted Actor-Learner. |
| **INCP** | In-Network Computing Provider. |
| **INSCSA** | Improved Non-Dominated Sorting Crow Search Algorithm. |
| **IoT** | Internet of Things. |
| **IoV** | Internet of Vehicles. |
| **IPGA** | Improved Parallel Genetic Algorithm. |
| **IPSO** | Improved Particle Swarm Optimization. |
| **IR** | Individual Rationality. |
| **ITS** | Intelligent Transportation Systems. |
| **IV** | Integrated Virtualization. |

| | |
|---|---|
| **LBL** | Load Balancing Level. |
| **LC** | Least Connection. |
| **LI-X** | Least Impact-X. |
| **LJFP** | Longest Job Fastest Processor. |
| **LLH** | Low-Level Heuristic. |
| **LRFC** | Learning Repository Fog-Cloud. |
| **MADRL** | Multi-Agent Deep Reinforcement Learning. |
| **MCEETO** | Multi-Classifiers Energy Efficient Task Offloading. |
| **MCS** | Minimum Cost Strategy. |
| **MCT** | Minimum Completion Time. |
| **MEC** | Mobile Edge Computing. |
| **MGAPSO** | Modified Genetic Algorithm and Particle Swarm Optimization. |
| **MIPS** | Million Instructions Per Second. |
| **MILP** | Mixed Integer Linear Programming. |
| **ML** | Machine Learning. |
| **MOPSO** | Multi-Objective Particle Swarm Optimization. |
| **MPSO** | Modified Particle Swarm Optimization. |
| **MUE** | Mobile User Equipment. |
| **NLP** | Nonlinear Programming Problem. |
| **Non-Convex Optimization** | An optimization problem where the objective function or constraints are non-convex, often leading to multiple local minima or maxima. |
| **NP-Complete** | Non-Deterministic Polynomial-Time Complete. |
| **NSGA-II** | Non-Dominated Sorting Generic Algorithm II. |
| **OptFogCloud** | Optimal Fog-Cloud Offloading Framework. |
| **PABP** | Penalty-Aware Bin Packing. |
| **PNN** | Probabilistic Neural Network. |
| **PORA** | Predictive Offloading and Resource Allocation. |
| **PPO** | Proximal Policy Optimization. |
| **PS** | Priority Scheduling. |
| **PSO** | Particle Swarm Optimisation. |
| **PTPNS** | Priced timed Petri nets. |
| **PTR** | Perception-Reaction Time. |
| **QoE** | Quality of Experience. |
| **QoS** | Quality of Service. |
| **RAM** | Random Access Memory. |

| | |
|---|---|
| **ReLU** | Rectified Linear Unit. |
| **RL** | Reinforcement Learning. |
| **RNN** | Recurrent Neural Network. |
| **Robust-CompOff** | Robust Computing Offloading. |
| **RP** | Resource Provisioning. |
| **RR** | Round Robin. |
| **RWRR** | Remind Weighted Round Robin. |
| **SA** | Simulated Annealing. |
| **SACO** | Smart Ant Colony Optimization. |
| **SDN** | Software Defined Networking. |
| **SFC** | Service Function Chaining. |
| **SJF** | Shortest Job First. |
| **SJFP** | Shortest Job Fastest Processor. |
| **SLA** | Service Level Agreements. |
| **SMA** | Stable Matching Algorithm. |
| **SMO** | Spider Monkey Optimization. |
| **SPEA-II** | Strength Pareto Evolutionary Algorithm-II. |
| **SPP** | Service Placement Problems. |
| **SRPM** | Service-Request Prediction Model. |
| **STLW** | Shorter Slack Time Less Remaining Workload. |
| **TCAS** | Time Cost Aware Scheduling. |
| **Telesurgery** | Refers to performing surgery remotely using robotic systems controlled by a surgeon through a high-speed internet connection. It allows specialists to operate on patients from a distance. |
| **V** | Vertex. |
| **VED** | Vickrey-English-Dutch. |
| **VM** | Virtual Machine. |
| **WA-FSP** | Whale Optimization. |
| **WAP** | Wireless Access Points. |
| **WI** | Whittle Index. |
| **WOA-FMP** | Whale Optimization Algorithm for Fog Module Placement. |
| **WRR** | Weighted Round Robin. |
| **5G** | Sixth-generation Mobile Networks. |
| **0-1 knapsack method** | A combinatorial optimization technique used in resource allocation |

# List of Mathematical Operators and Notations

| | |
|---|---|
| $PBest_{id}$ | Best Particle Position (Local Best) |
| $gBest_{id}$ | Best Group Position (Global Best) |
| $\boldsymbol{b}$ | Bias. |
| $DS_i$ | Data size |
| $CC$ | CloudSim Clock |
| $ET$ | Emitting time of the tuple |
| $w_k, w$ | Objective weights |
| $SV_n$ | Number of Service |
| $\|M_{threshold}\|$ | Maximum number of active nodes allowed at any time |
| $x_n$ | A Binary Indicator for whether fog node n is active or not |
| $y_{n,s}$ | A Binary Indicator for whether sensor n is associated with fog node n |
| $\|M\|$ | Total number of fog nodes |
| $\|S\|$ | Total number of sensors |
| $\|Sn\|$ | Maximum number of active sensors managed by each fog node |
| $\lambda$ | Average Latency of Tasks |
| $N^T$ | Total Number of Tasks |
| $A_y$ | User Application |
| $R_{a,j}$ | Available resource at fog node $F_j$ |
| $R_{tot,j}$ | Total resource of fog node $F_j$ |
| $R_{u,j}$ | Resource currently used at the fog node $F_j$ |
| min | Minimum Function |
| $\sum$ | Summation function |
| $\forall$ | Universal Quantifier, Means For All |
| s.t. | Subject to |
| $S$ | Sensor |
| $\in$ | Element of |
| $D_{\mathbf{avg}}$ | Average Loop Delay |
| $U_{\max}$ | Maximum Bandwidth Utilization |
| $c1, c2$ | Random Parameters Between 0 and 1 |
| $r1, r2$ | Positive Constants |

| | |
|---|---|
| $Th_{s,j}$ | Transfer rate |
| $L_{i,j}$ | Total delay |
| $D_{pn}$ | Propagation delay |
| $D_{pr}$ | Processing delay |
| $D_q$ | Queuing delay |
| $D_{tx}$ | Transmission delay |
| $B_{i,j}$ | The amount of Bandwidth |
| $P_{i,j}$ | Power consumption |
| $E_{i,j}$ | Energy consumption |
| $\bar{L}$ | Average delay |
| $A$ | Set of application $A = A1, A2, ...., A_N$ |
| $F$ | Set of fog nodes $F = F1, F2, ...., F_M$ |
| $T_{Ay}$ | Set of tasks belonging to applications $A_y$ |
| $i$ | Index for tasks |
| $j$ | Index for fog nodes |
| $y$ | Index for application |
| $x_{ij}$ | A Binary indicator with either 0 or 1 Value |
| $R_{\mathbf{Req}}$ | resources Requirements for task $T_i$ |
| $Q_j$ | Current workload at fog node $F_j$ |
| $C_j$ | Capacity of fog node $F_j$ |
| $D_{ij}$ | Distance between task $T_i$ and node $F_j$ |
| $L_{ij}$ | Total delay in task $T_i$ and node $F_j$ |
| $\bar{L}$ | Average delay across all tasks |
| $E_{ij}$ | Energy consumption |
| $R_{i,j}$ | Resource cost for task $T_i$ |

# Chapter 1

# Introduction

## 1.1 Background

In recent years, the IoT has changed the world, with over 20 billion devices connected to the Internet. Researchers believe that the number of these devices will increase to 75 billion by 2025 [1, 2]. This dramatic change has influenced both business vision and how we interact with technologies. IoT applications are considered to be diverse; including smart home, smart city, healthcare, and industrial IoT. In smart homes, IoT plays a vital role because it includes connected thermostats and sensors. In smart cities, there are IoT devices everywhere, e.g. traffic sensors, environmental sensors, and utility sensors. Elsewhere, healthcare and industry IoT works in many different ways, e.g. remote patient monitoring, emergency response, supply chain optimization, and predictive maintenance.

However, with this remarkable growth in IoT, significant challenges arise, such as latency, security, and privacy concerns, which stakeholders must address to overcome this dilemma [3, 4]. IoT and portable technologies are digital identification tools that enable end-users to collaborate, integrate and manage computing resources. These devices lack the resources to handle the immense data generated by their applications [5]. Though there are several cloud-based solutions available to address limitations, but these solutions may have drawbacks such as latency, security, privacy, and protection, which can potentially weaken digital operations and real-time applications.

More precisely, a cloud is a centralised entity, whereas IoT and portable devices are distributed across diverse locations. Thus, delay-sensitive applications cannot

Portions of this chapter are sourced from the paper listed in Publication Section as: *Resource Scheduling in Integrated IoT and Fog Computing Environments: A Taxonomy, Survey and Future Directions*

reliably access cloud resources because of high network traffic, significant bandwidth requirements and the requirement for permanent Internet connectivity. To address this, middleware solutions emerge as a viable approach to this problem, by bringing cloud services closer to end devices[6]. These middleware solutions primarily involve distributed computing paradigms such as cloud, fog and edge computing. Each of these paradigms addresses IoT challenges differently by varying proximity, resource allocation methods, and architecture, providing tailored solutions that significantly improve performance, security, and real-time responsiveness.

## 1.2 Cloud Computing: Overview and Challenges

The term cloud computing refers to processing data through the Internet, providing tremendous storage along with programs to process data, allowing users access from a remote location. Enterprises offer computing services to their clients with the help of cloud computing, one of the most popular models is the pay-as-you-go service, reducing the cost burden [2]. The back-end section consists of databases, storage, computers, and servers interconnected to create a cloud network. Cloud computing operates if Internet connectivity is available and applied to numerous domains [7].

However, IoT applications face challenges with latency, bandwidth, security and privacy due to centralised computing far from IoT devices [8]. Resource allocation between clients and servers is extremely important; it manages, controls, and monitors resources to increase system stability and improve user experience. It balances the system load by reducing execution time based on task requirements [9]. Resources are provided from a resource pool using virtualisation, enabling a single physical instance to serve multiple clients simultaneously, enhancing CPU and disk utilisation. Commonly, tasks from clients are offloaded to the cloud, creating significant resource management challenges due to increasing complexity and dynamic resource demands. Challenges include:

- Task execution delay
- Multiple resource allocation
- Task dependencies
- Load balancing
- Exorbitant cost
- Service quality and user experience

Despite many benefits, cloud computing also has challenges and limitations, making risk assessments beneficial before adopting cloud solutions. Public clouds offer generic, multi-tenancy services that may not align with specific organisational processes.

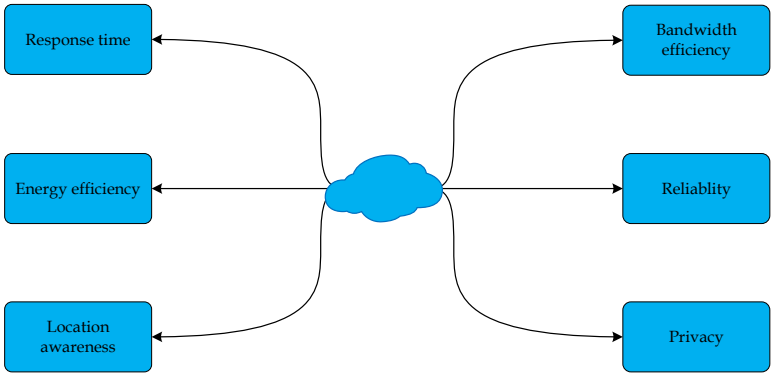


Figure 1.1: Challenges of Cloud Computing

Figure 1.1 presents several challenges that can affect performance and operations in cloud computing. Response time can be slow due to the distance between users and data centres, which can impact real-time applications. Bandwidth efficiency can be a concern when high data transfer requirements strain network resources, leading to costly solutions. Although rare, reliability issues can lead to service disruptions or downtime, which are especially concerning for mission-critical services. Privacy is also a significant concern as data is stored remotely, requiring trust in cloud providers and compliance considerations. The lack of location awareness can hinder optimisation in applications that rely on geographical data. Furthermore, energy efficiency is crucial as cloud data centers consume a substantial amount of energy, necessitating eco-friendly and cost-effective solutions. These factors highlight the complex landscape of cloud computing, where careful management is necessary to mitigate their impact.

## 1.3 Edge Computing: Overview and Challenges

Edge computing is a distributed computing paradigm that brings data processing, computation, and storage closer to the devices generating data, such as IoT sensors

and actuators. This proximity to the data source reduces latency, improves response time, and optimises bandwidth by minimising data transmission to the centralised cloud servers. It is particularly beneficial for real-time applications, such as autonomous vehicles and smart cities, where quick decision-making is critical [10, 11].

These are the main characteristics of edge computing [12, 13].

- **Decentralized Resource:** Computation and data storage are performed on either local devices or servers.
- **Low latency:** Provides extremely low latency, which is critical for real-time applications.
- **Bandwidth:** The bandwidth is less than that of other cloud computing solutions because it reduces the amount of data transmitted to the cloud.
- **Autonomy:** Devices have the capability to operate independently, even with intermittent internet connectivity.

Dispute offering benefits, Edge computing encounters challenges such as limited resources and complex management. Here are the main drawbacks of edge computing

- **Resource constraints issues:** Edge devices have limited computational power, storage and memory compared to other technologies, which limits their ability to handle complex tasks. Also, it is challenging to handle large datasets.
- **Security risk issues:** Managing security across many devices and locations can be challenging, which leads to a potential risk of data breaches or cyberattacks.
- **Cost and management Complexity issues:** Managing and maintaining a large number of edge devices can be complex and costly. A strong infrastructure and management tools are important to ensure smooth operation, updates, and troubleshooting.
- **Interoperability issues:** Integrating heterogeneous devices or different devices from different vendors can be a challenge to ensure smooth operation. This issue can have a direct impact on deployment and scalability.
- **Resource management issues:** Handling and storing a large amount of data at the edge can be a real challenge. Many applications require a large amount of data to be processed and stored locally, which is difficult in an edge computing environment.

- **Latency and energy consumption issues:** Using edge computing aims to reduce latency, but it still depends on the network conditions. Poor network connectivity can disrupt the performance and dependability of applications that utilise edge computing [14, 13].

## 1.4 Fog Computing: Overview and Architecture

Scholars have introduced several technologies to overcome the limitations of cloud computing; however, these solutions violate QoS, such as latency, response time, security and privacy. Fog computing is one of these solutions that has attracted researchers' attention to optimise the quality of service. Fog computing is a decentralised, proximity-based solution that provides computing services to clients at distributed locations [15, 16].

The term fog computing does not replace the term cloud computing. It only extends the cloud resources by offloading the computation, data and application on the edge of the network in the proximity of the end-user instead of the cloud [17]. This enables service providers to take advantage of the clients' devices at some distributed locations, which efficiently enhances the utilisation of resources. As shown in Figure 1.2, this technology resolves the issue of delay by reducing computing in the cloud. At the same time, it also addresses bandwidth usage because the majority of computing happens near the end user instead of the cloud itself, which will reduce network traffic.

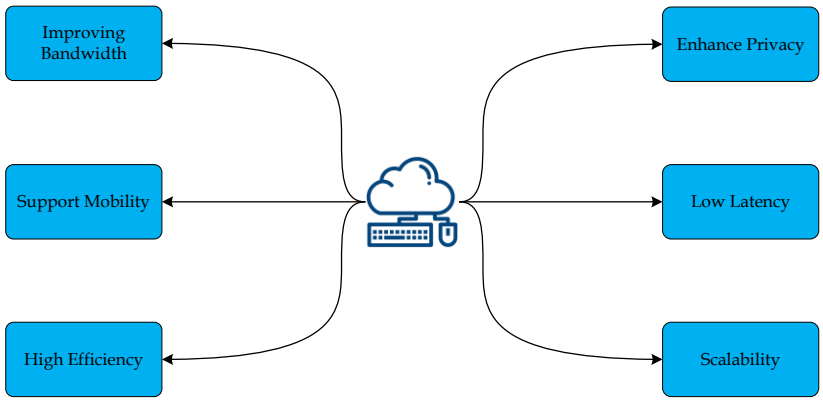


Figure 1.2: Advantages of Fog Computing for Resource Scheduling Strategies

### 1.4.1 Architecture and Key Concepts

Fog computing was introduced by Cisco in 2012 as a decentralised computing system [18]. It is a distributed computing paradigm that extends cloud services to the network edge, enabling local data processing and storage to support IoT applications, reducing latency and bandwidth usage. Resource scheduling is implemented at a fog network, which resolves the network issues and improves the network performance by means of resource utilisation that is standardised, adaptable and equitable [5]. Figure 1.3 shows the layered architecture of a fog computing environment.

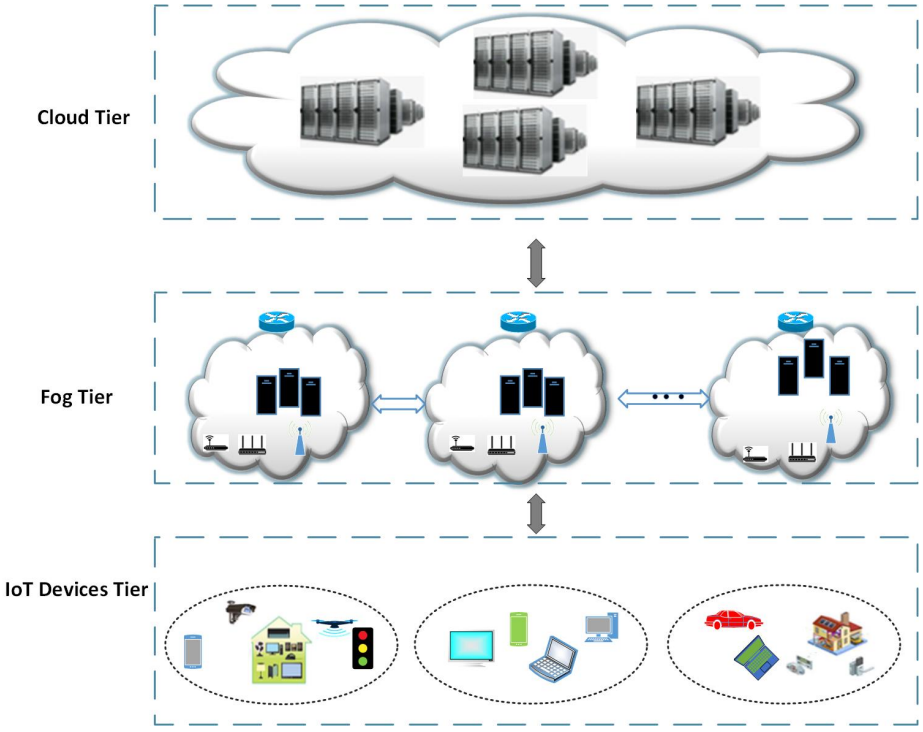


Figure 1.3: Fog Computing Architecture

#### IoT Layer

This layer represents IoT devices, such as smartphones, drones, smart vehicles and sensors. Those devices are usually distributed in different regions and places. The scope of these applications incorporates a wide range of industries and sectors, including the health sector, emergency management, smart homes, telecommunications, traffic systems, agriculture, aerospace, intelligent transportation, hospitality, food, water supply, organisational maintenance and disaster management, among many

others. Conventional assumptions may consider that the tasks belonging to delay-sensitive applications are provisioned in the computing resources at the fog level [5]. IoT devices typically generate tasks that can be delay-sensitive or resource-intensive, requiring further processing at higher layers.

#### Fog Layer

This layer can have a large number of nodes that have fog computing services. Fog nodes are logical and physical network elements that can do computing, store data temporarily and transmit data between the cloud and end-user devices (IoT layer). Moreover, fog nodes can be either resource-poor devices or resource-rich machines. Resource-poor devices include wireless access points (WAP) or basic routers with limited capabilities. On the other hand, rich devices include cloudlets, which act as mini-cloud data centres providing substantial processing resources for IoT applications [2].

Due to the resource constraints in the IoT layer, fog nodes enable these devices to share and access their desired computing resources to satisfy the task requirements needed to perform their intended operations. Resource scheduling is supposed to be proposed at this layer to maximise the best possible use of the computing resources for client applications. Hence, in order to ensure QoS, it becomes significant to devise a resource scheduling scheme. Since IoT-based applications generate tasks in a fog computing system that are heterogeneous, organised and controlled by the client devices at distributed locations, the nature, type and characteristics of these devices must be incorporated while allocating resources for the fog computing paradigm that would result in relatively well-organised task execution that is also more efficient.

#### Cloud Layer

This layer represents a top-layered stage, which has traditional centralised cloud resources with extensive storage capabilities. Some cloud services are shifted to the fog layer to reduce delay, improve response times, and enhance security by decreasing the load on cloud resources.

### 1.4.2 Comparative Analysis of Cloud, Edge and Fog

Table 1.1 provides a side-by-side comparison of the most prominent distributed systems, including Cloud, Fog and Edge. This summary discusses the advantages and challenges of each environment.

| **Aspect** | **Cloud Computing** | **Fog Computing** | **Edge Computing** |
|---|---|---|---|
| Architecture | Centralised data centres and servers | Decentralised with intermediary nodes closer to the edge | Highly decentralised, processing at the source. |
| Resource Constraint | No significant resource constraints, high computational power | Distributed resources, better load balancing | Limited computational power and storage. |
| Security | Strong centralized security, but vulnerable in transit | Enhanced security through local encryption | Security can be high with proper encryption, but device heterogeneity increases attack surfaces. |
| Management | Centralized management, easier to maintain | Distributed management,requires sophisticated tools and automation | Complex management and maintenance,dispersed devices. |
| Scalability | Highly scalable with large resources | Moderately scalable, but limited by node capacity | Less scalable, limited in remote areas, network latency. |
| Mobility | Not Supported | Supported | Supported. |
| Connectivity | Dependent on internet, high bandwidth costs | Moderate dependency, improved by local nodes | Low internet dependency, but connectivity issues may arise in highly remote areas. |

Table 1.1: Comparison of Cloud, Fog, and Edge Computing:Advantages and Challenges [19, 10, 13]

The hierarchical architecture in Figure 1.4 presents the physical and logical distance between various distributed systems and the edge devices. These edge devices include IoT sensors, actuators, and smartphones that initiate requests. The closest data processing to edge devices is edge computing, which performs localised processing through either edge servers, smart gateways, or even within the edge devices themselves. It enables real-time responsiveness with minimal delay. On the other hand, fog computing provides intermediate processing capabilities via fog nodes that are positioned closer to the edge. At the furthest point lies cloud computing, which has tremendous centralised data centres and advanced machine

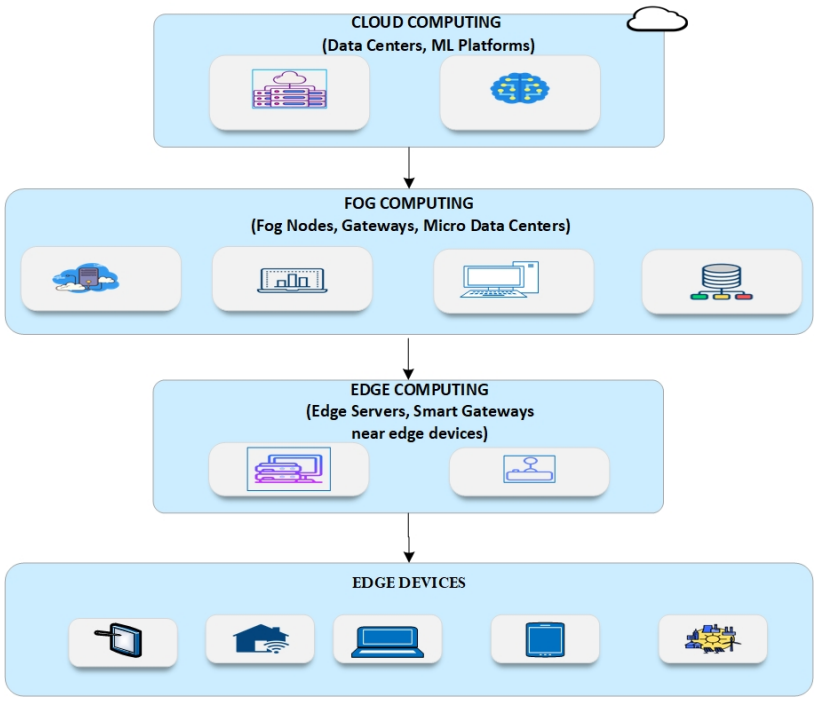


Figure 1.4: Hierarchical architecture illustrating the spatial distribution of data processing across Edge Devices, Edge Computing, Fog Computing, and Cloud computing layers

learning platforms with a huge capacity. The main dilemma of cloud computing is its distance from the edge devices, which can affect performance compared to edge and fog computing. By examining both the advantages and disadvantages of cloud and edge computing, it is clear that fog computing can offer a balanced approach and cover the limitations between cloud and edge technology. Its intermediary role provides enhanced security, lower latency, and better resource management, which makes it a compelling option for various applications.

## 1.5 Motivation

Fog computing offers several advantages, known as many-fog nodes with heterogeneous resources and limited computing storage capacity. Although the fog computing framework has attracted the attention of researchers and industry, it faces several critical issues in optimising QoS through resource management approaches. Enhancing the QoS will significantly impact smart environments, including the smart healthcare sector, smart cities, smart homes, smart factories, agriculture and the IoT

industry.

From the perspective of the smart healthcare sector, delays are critical, especially for emergency situations, where even a second can save lives. A prime example is remote surgery "Telesurgery", which requires minimal delay with a high response time. Research has shown that a delay of 200 ms can harm patients in complex remote surgeries [20]. The optimal delay recommended by another research should be less than 100 ms [21] . Both of those scholars emphasise the importance of delay parameters in remote surgery. Also, delay plays a major role in real-time patient monitoring.

In smart cities and smart homes, QoS is also important to ensure stability of life. In smart cities and smart grids, sustainability is a real concern for most stakeholders, who will focus on the importance of efficient resource management, including bandwidth utilization . With the increase of wearable technology and IoT devices, the demand for high bandwidth and reliable connection is essential to maintain the functionality of smart cities [22, 23].

From a smart factory perspective, reducing energy consumption helps to lower $CO_2$ emissions and operational costs. Secure and efficient data processing ensures timely decision-making, enhancing system reliability [24]. From an agriculture perspective, QoS plays a vital role in optimizing resource management by ensuring reliable data transmission, and enhancing decision-making processes. This field is witnessing a significant increase in IoT device adoption for real-time monitoring of soil, weather and crops, while maintaining a high network availability and low latency. Optimizing bandwidth utilization and secure data processing supports precision agriculture, reducing costs and improving sustainability[25, 26]

## 1.6 Research Questions

1. What are the critical limitations of existing resource management techniques in the fog computing environment, and how can these limitations be addressed to optimise resource usage and improve QoS?

2. How can a GGCN-based resource scheduler effectively optimise resource allocation and the number of active fog nodes in fog computing environments to achieve minimal loop delay and reduce network usage while outperforming existing methods?

3. What innovative approaches can an eMLP-based task scheduler offer to minimise delay, lower resource costs, and optimise the number of fog nodes and total network usage in fog computing?

4. How do advanced scheduling strategies impact QoS in dynamic and heterogeneous fog computing environments, and what metrics best demonstrate their efficiency?

## 1.7 Contributions and Methodology

This thesis makes the following significant contributions:

- **A qualitative comparative analysis** that classifies existing fog resource management approaches into task-oriented and resource-oriented categories, providing a structured perspective on current techniques.
- **A Novel Resource Scheduling model** based on gated graph convolutional neural networks (GGCNs), which balances average loop delay and total network usage while minimising the number of active nodes.
- **A multi-objective scheduling framework** utilising enhanced multi-layer perceptrons (eMLP) to optimise key QoS metrics, including delay, cost, and power consumption, considering resource demands to prevent over-provisioning and under-utilisation.

The methodology adopted to achieve the above contribution are follows:

- Comprehensive Literature Review
  (a) Conducted a systematic review of existing literature on resource management in fog computing environments.
  (b) Categorised these approaches into key aspects of resource management (e.g., task scheduling, application placement).
  (c) Identified the current gaps and outlined research directions.
- Development of GGCN-based resource allocation
  (a) Designed a GGCN-based model to optimise resource allocation.
  (b) Defined the constraints for resource allocation, including sensor activity, resource availability and fog node availability.
  (c) Implemented and tested the model in a simulated fog computing environment.
  (d) Benchmarked the GGCN-based scheduler against PSO, SJF, and FCFS algorithms to evaluate its performance in terms of QoS metrics such as loop delay and network usage.

- Implementation of eMLP-based Task Scheduling
  (a) Designed an innovative eMLP-based task scheduler model to minimise delay and resource costs.
  (b) Developed a simulation framework to evaluate task scheduling in fog environments.
  (c) Compared the performance of the eMLP-based scheduler against other approaches (GCN, FCFS, SJF) using QoS metrics such as delay and network Usage.

## 1.8 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 presents an overview of fog computing, followed by a comprehensive literature review of resource management approaches in fog computing to optimize QoS, with a focus on task-oriented management approaches, based on the work published in iv. Chapter 3 discusses advanced resource management approaches, focusing on resource-oriented strategies. Additionally, it provides an overview of the simulation tools for computing environments. In chapter 4, optimising delay and total network usage with GGCN in fog computing, based on the work published in iv. Chapter 5, introduces a novel task scheduling approach using the eMLP mechanism, which optimizes multiple QoS parameters, including delay, network usage, power consumption, and cost. Finally, chapter 6 concludes the thesis and outlines future directions, emphasising the potential enhancement of QoS optimisation in fog computing systems.

# Chapter 2

# Task-Oriented Approaches

## 2.1 Motivation

The motivation behind this chapter is to provide a qualitative comparative analysis of resource management approaches. It specifically focuses on task-oriented approaches for optimising QoS in fog computing environments. Although fog computing was introduced several years ago, it continues to attract significant interest for researchers and industry experts. Due to the limited resources at the fog node, effective scheduling mechanisms such as task scheduling, task offloading, and application placement are essential for improving system reliability and performance.

It will also motivate future researchers to identify the research gaps and provide a clear picture of future directions. This review classifies resource management into up-to-date categories to assist future researchers in optimising QoS. Moreover, most existing literature reviews are either outdated or narrowly focused on one or two categories in resource management, neglecting others.

## 2.2 Resource Management in Fog Computing

This chapter provides a qualitative comparative analysis of resource management approaches and their limitations. Resource management approaches in fog computing can be classified into six main categories: application placement, task scheduling, resource allocation, task offloading, load balancing, and resource provisioning. However, this chapter highlights the task-oriented management approaches for

Portions of this chapter are sourced from my published paper: *Resource Scheduling in Integrated IoT and Fog Computing Environments: A Taxonomy, Survey and Future Directions*[27].

optimising QoS in fog computing. These approaches are application placement, task scheduling and task offloading. Despite its advantages, fog computing introduces weaknesses that encourage scholars to investigate resource management approaches, including:

- Cost and Management Complexity Issues: High deployment costs for hardware (such as routers, gateways, hubs) and complex system management due to decentralised nodes.
- Security Risk Issues: The distributed architecture heightens the risk of cyberattacks, data breaches, and privacy concerns.
- Resource Constraint issues: Fog nodes have limited storage capacity, computational power, and energy compared to cloud data centres, which limit their operational efficiency.
- Resource Management Issues: Managing multiple fog nodes in a geo-distributed setting presents challenges that require advanced orchestration and monitoring solutions.
- Interoperability and Standardisation Challenges: Ensuring seamless communication between diverse fog nodes, IoT devices, and cloud systems remains difficult due to a lack of standardisation [18, 7, 10].

This chapter will focus on task-oriented management strategies to optimise QoS parameters in fog computing. It seeks to clarify critical issues and highlights the significant studies of application placement, task scheduling and task offloading- each of which is examined in the following subsections.

### 2.2.1 Application Placement

Application placement refers to applications that require more than one service, such as those in the smart industry, smart city, or smart vehicle. This occurs when IoT devices in these applications require more than one service, and these services should be mapped to one or more fog nodes. Assume application request $n$ services $\{SV_1, SV2_2, SV_3, \ldots, SV_n\}$ a number of services and should be mapped to available m fog node $F = \{F_1, F_2, F_3, \ldots, F_M\}$. According to available resources, each available fog node might serve more than one service.

Here are the three critical challenges faced by scholars in the application placements approach.

1. **Resource Utilisation:** The main dilemma is to ensure that fog nodes are utilized effectively without underuse or overload between them.

2. **QoS Requirements:** It is essential to meet specific criteria like service deadlines, latency, and reliability, which are critical to the performance of IoT applications.

3. **Dynamic environment:** Fog computing is unpredictable and highly variable, with changes in the network conditions based on resource availability and workload demand.

**Application Placement in Practice:** Consider a smart city application with various IoT services. These services include pollution detection, traffic monitoring and emergency response, which are required to be deployed on fog nodes across the city. The application placement issue includes:

- Determining which available fog nodes can host these services.
- Ensuring the placement strategy does not violate latency requirements for real-time monitoring and response.
- Ensuring the load is balanced across fog nodes to avoid overuse and congestion.

**Problem definition for application placement approach**.

- **Objective:** To find an optimal placement solution that will map IoT services to available fog nodes. The optimal solution has to take into consideration QoS requirements and optimise objective functions such as latency, energy consumption or cost.
- **Mapping:** The solution includes mapping each IoT service to one or more fog nodes, which means each fog node can host multiple IoT services.

Figure 2.1 classifies the application placement approaches based on optimisation strategies into Quantitative Optimisation Techniques, Heuristics, Metaheuristics, and AI-based Approaches.

### Heuristic Approach

Several scholars contributed to application placement in the heuristic approach to optimise QoS. Selimi et al.[28] introduced a heuristic placement algorithm called the Bandwidth and Availability-aware Service Placement (BASP) algorithm to improve service placement in Community networks (CNs). The algorithm focuses on optimising the deployment of services within micro-cloud environments such as Guifi.net. BASP aims to enhance the quality of experience (QoE) for users by optimising bandwidth (achieving 2x bandwidth gain) and reducing the response time (37% reduction in packet loss rate). The proposed algorithm was tested with

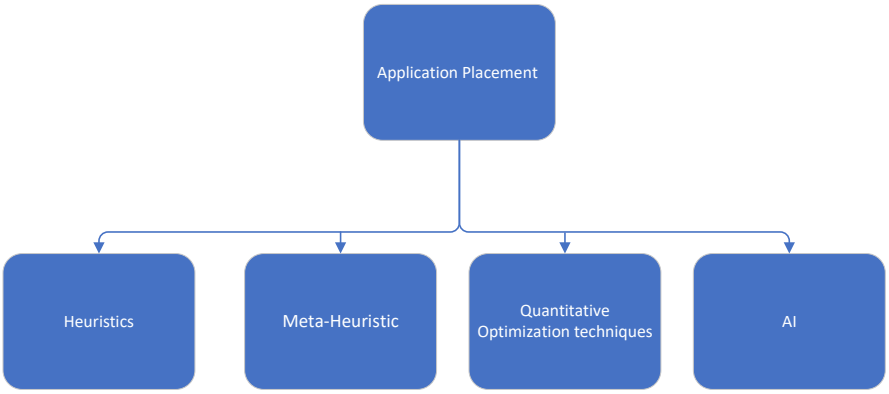


Figure 2.1: Taxonomy of Application Placement Approaches

services like live video streaming and Web 2.0 services and evaluated in a real CNs environment. Mahmud et al. in [29] investigated how to optimise QoS by managing application modules based on their latency sensitivity. The approach was applied in a smart home scenario via IfogSim simulation. The paper proposed two algorithms:

- The first algorithm is applied to the placement of application modules. It aims to ensure modules are placed in appropriate fog nodes based on several factors, including delay and resource availability.
- The second algorithm focuses on resource optimisation by forwarding these modules. It aims to forward application modules from underutilised nodes to more highly occupied ones to reduce the number of active nodes.

The study addresses both latency-sensitive and latency-tolerant applications, prioritising the placement of the more sensitive ones. The outcome of this study is reducing delays by ensuring the deadline satisfies QoS while also optimising resource use by minimising the number of active nodes.

Mahmud et al. [30] presented a new strategy for energy-aware applications deployed in fog environments. Their approach is called an energy-aware allocation strategy, based on placing modules on suitable fog devices instead of using the cloud. They applied their strategy in a healthcare scenario in the IfogSim simulation. The results of their experiment are reducing energy consumption by 2.7% compared with the cloud and 1.6% compared with the default fog configuration. It also reduces the average delay by almost 0.5% and 18%, respectively. In network usage, both the proposed strategy and the default fog configuration performed equally but showed notable improvement compared to cloud-only by nearly 95%. Taneja and Davy [31] proposed a new model mapping algorithm which will be able to distribute the

computation process between fog and cloud layers dynamically. Their approach is designed to reduce the application latency and energy consumption and optimise network usage. It was implemented in IfogSim simulation in a generic scenario.

Mann [32] discusses the challenges of application placement in fog computing and proposes his solution to overcome these challenges. As it is known, fog computing includes placing applications in various resources such as fog nodes and cloud data centres. The consequence of the placement of these applications negatively impacts the performance metrics, including latency and bandwidth. The main challenge addressed by the author was the scalability issues; the paper proposes a decentralised optimisation technique where the infrastructure is divided into fog colonies. It is based on a group of computational resources in a certain geographical region. Each application placement is performed separately for each fog colony. However, independent optimisation gives a sub-optimal solution due to missed synergies between fog colonies. Therefore, some coordination between these fog colonies might be beneficial to improve QoS. In his proposed model, an ILP-based algorithm is used to find an optimal solution to determine where to host each application component and how to route connectors. Also, a heuristic search-based algorithm is used to construct a solution. The proposed approach shows better results compared with the other four different approaches. The first approach is centralised decision-making, which is performed for the entire infrastructure at once. The second approach is independent fog colonies, where each fog colony is optimised independently. The third approach is fog colonies with communication, where excess application components in one fog colony can be offloaded to neighbouring fog colonies. The fourth approach is fog colonies with overlaps, where shared resources can be dynamically allocated between neighbouring colonies.

Oliveira et al. in [33] proposed a method to tackle application placement issues, especially for applications that are latency-sensitive and communication-intensive. The paper discusses a hierarchical fog computing model with several levels. Each level comprises cloudlets with different computational resources and connectivity. Applications are divided into modular parts, allowing each module to be executed on different devices in the fog hierarchy. The problem is determining the optimal placement of these modules to meet application requirements and optimise QoS. The authors introduce the Least Impact -X mechanism (LI-X) to optimise the placement application. LI-X calculates the average MIPS required for each model to perform the resource allocation. At first, LI-X allocates the resources in the current cloudlet. If the resources are insufficient, it identifies the modules that can be moved to a higher level with minimal communication impact. The main aim of the proposed algorithm is to utilise resources at the lower fog level as much as possible. In the evaluation, four scenarios were considered:

1. Topology A, a single fog level.

2. Topology B, a single fog level with increased distance between devices.
3. Topology C, two fog levels.
4. Topology D, with three fog levels.

The outcome of their investigation shows that LI-X has better results compared to existing benchmark schemes, including DP-I and CB-E, in terms of network traffic, response time and Module distribution in all Topologies.

**Meta-Heuristics Approach**

In the Meta-heuristics approach, several researchers applied different algorithms for optimisation purposes. Lin and Yang [34] addressed computation requirements by IoT devices in Industry-4.0 logistic centres. The Authors Proposed a new approach to solve this NP-hard problem by using DMGA, which is a hybrid combination of Discrete Monkey (DMA) and Genetic Algorithms (GA). Their proposed system model deals with static sensing devices that gather data from fixed positions in a logistics environment. Here are the functionalities of these combinations

- DMA algorithm: The core mechanism used to optimise the placement of fog, edge, and gateway devices.
- GA algorithm: Incorporated into DMA to enhance computational efficiency. DMGA utilises crossover and mutation operations of GA, which help to explore the solution space effectively and speed up convergence towards the optimal solution.

Their objective is to find the optimal deployment of a fog system in a logistics centre to enhance the performance of Industry 4.0. The simulation proved that their proposed algorithms were able to minimise installation costs while adhering to performance standards compared to GA and MA. Natesha and Guddeti [35] proposed a metaheuristic approach called the NSGA-II algorithm for data replica placement to enhance data-intensive IoT applications. The objective of this study is to reduce latency and data access costs while increasing data availability. The NSGA-II algorithm is used to optimise the placement of data replicas closer to users. The system model includes three layers of typical fog computing and a data replica manager (DRM), which play vital roles in monitoring fog node conditions and data requests. Their system model focuses on the fog layer, which is divided into:

- Fog node: Classified into domains, each domain contains a regional fog node (RFOG) with sub-local fog nodes (LFOG).
- DRM: It has three main functionalities:

1. Monitoring IoT request access.
2. Predicting the number of data replicas required for IoT.
3. Optimising data replica placement by finding the optimal nodes using the NSGA-II algorithm.

The NSGA-II algorithm predicts and places replicas based on QoS requirements. The simulation results show that NSGA-II has better results than FFRPP and MCS in improving QoS. Natesha and Guddeti [36] proposed two combined metaheuristic mechanisms to tackle service placement issues in a fog computing environment. The authors used the MGAPSO algorithm in their proposed solutions, which combines GA and PSO to evaluate fitness, perform selection, crossover, mutation, and optimise service placement iteratively. Their proposed model is built on a two-level fog computing architecture designed for IIoT applications. Authors classify the fog architecture into either

- Fog Master Node: Responsible for monitoring and controlling the Fog Cells and allocating resources optimally across Fog Cells using MGAPSO and EGAPSO algorithms. It acts as a central coordinator for the fog environment, ensuring efficient application placement. The Fog Master Node consists of four components:

  1. Fog Cell Registry: Registers and maintains information about available fog cells in the system. The registry allows Fog master to track the available and suitable fog cells for process tasks.
  2. Host Monitor: Monitoring resource usage in fog cells and making sure all resources are efficiently utilised.
  3. Fog Cell Controller: Maintaining the topology of the fog environment by monitoring the status of fog cells. It helps to get real-time updates about fog cells' condition and ensure that they are operational and connected.
  4. Fog Service Registry and Service Allocation: Handles the registration of services and allocates them across fog cells based on current resource availability and optimization algorithms.

- Fog Cells: contains several independent fog nodes with different capabilities, which act as an interface to IoT devices (sensors, actuators). All IoT requests will be processed in fog cell nodes except for complex or further processing, which will be forwarded to the fog master node. Fog cells act as the primary data processors, handling most of the computational load locally.

The authors also use EGAPSO, which enhances MGAPSO by incorporating elitism in GA to save the best chromosomes for the next generation without crossover

and mutation. This technique will speed up the convergence of EGAPSO. Their experiment results show that EGAPSO outperforms existing algorithms, such as FF, B&B and DEBTS, in optimising several QoS parameters. In paper [37], the authors formulate the service problem as a multi-objective optimisation problem and prove it to be NP-complete. The authors propose a combination of meta-heuristic algorithms: Genetic algorithms, Simulated Annealing (SA) and Particle Swarm Optimisation to optimise service placement. In their proposed model consists of multiple fog domains (fog nodes). Each domain has a fog control manager (FCM) to allocate the resources and service placement within the domain. The FCMs can monitor node status, handle incoming tasks, and communicate with other FCMs to ensure optimal application placement across domains. The GA-SA hybrid utilises GA for population initialisation and SA for local search to enhance the exploration capability. The GA-PSO hybrid combines GA's crossover and mutation operators with PSO's particles to update rules to balance exploration and exploitation. The results show that GA-SA hybrids (referred to as FSPGSA) outperform other algorithms, including GA-PSO, RFSP and FSPSA, in terms of makespan, energy consumption and cost. The results highlight the ability of GA-SA to manage service placement across multiple fog domains.

In Paper [38], the authors address the application placement issue via Enhanced Multi-Objective Particle Swarm Optimisation with Clustering (EMOPSOC). The EMOPSOC combined with the Fog Picker algorithm. The Fog Picker allocates IoT components to fog nodes based on QoS attributes, whereas EMOPSOC dynamically schedules fog nodes. Their system model utilises the EMOPSOC algorithm to evaluate fitness and performs non-dominated sorting, Clustering by K-means, crowding distance calculations, and Euler distance computation to optimise scheduling. The functionality of the Fog Picker is to use a weighted vector approach to map IoT components to the most suitable fog nodes. The results prove that EMOPSOC with Fog Picker improves resource scheduling and application placement compared with other algorithms, including MOPSO, NSGA-II and SPEA-II, in terms of steady performance and minimising resource wastage.

### Quantitative Optimisation Approach

In the Quantitative Optimisation Techniques (Mathematical Programming) approach, several researchers have proposed models to offload tasks and optimise QoS. For example, Velasquez et al. [39] propose a modular architecture for intelligently placing IoT services to reduce service latency. The architecture consists of three components: a service repository, an information collection module, and a service orchestrator. Integer Linear Programming (ILP) is employed within the service orchestrator as a decision-maker to optimise placing the service of tasks based on the network conditions. The paper discusses several applications within smart city including,

smart buildings, smart energy grids and smart mobility solutions. Bellmann et al. [40] deployed the Markov Decision process to optimise data replication in a fog computing environment, aiming to achieve low-latency data access for mobile clients. Their use of Markov model algorithms aims to keep client-specific data in the closest fog node without incurring the overheads of global replication. According to their simulation results, data availability is improved by 35% compared to the default replication strategies. The proposed model is based on the Fusion Multi Order Markov Model(FOMM), a combination of multiple time discretisation techniques and variable-order Markov models to predict the next node the client will be connected to. This allows proactive replication while minimising excess data. In their simulations, the proposed model shows notable improvements over baseline approaches like reactive replication in terms of reducing storage and communication.

Baranwal and Vidyarthi [41] proposed a new model to enhance application placement, called the Fog Orchestrator Node Selection (FONS) model. The study focuses on selecting the most suitable FONs based on identified performance metrics like the energy consumption or battery power of Fog Gateway Nodes(FGNs). The FONS model itself might be placed on either fog or cloud nodes, depending on predictive mechanisms. The model comprehensively considers the requirements of IoT devices, FGN resources, and the connected Fog Computational Nodes(FCNs), evaluating metrics like average processing speed and resource availability. In the paper [42], Aldossary proposes a new mechanism to optimise the placement of IoT applications using mixed-integer linear programming (MILP). The proposed mechanism is designed for a smart city scenario with several objectives, including minimising power consumption, data transmission, and associated costs. The model utilises both fog and cloud resources, factoring in node geographical locations, resource availability, and capacities. The mechanism optimises IoT application placement by balancing these constraints to ensure efficient use of energy and resources. It considers the IoT application requirements, available resource capacity, and geographic locations of servers. The simulation results show that MILP performance is better than the cloud-only approach and the heuristic algorithm in optimising QoS.

### AI Approach

In the AI approach (machine learning, deep learning, neural networks, reinforcement learning), several researchers have applied approaches in machine learning (deep learning, reinforcement learning, federated learning) and neural networks. Goudarzi et al. [43] applied a deep reinforcement learning mechanism (DRL) in a fog computing environment to enhance the performance of the application via the application placement approach. The approach called experience-sharing application placement XDDRL helps to tackle complex and dynamic IoT applications modelled (directed

acyclic Graph (DAGs)). Their study claims that DAGs are modelled as IoT applications, and they used a framework called importance-weighted actor-learner (IMPALA). Additionally, they introduce X-DDRL, an experience-sharing distributed framework to allow several fog brokers to interact with the fog environment in parallel, reducing exploration costs. They use cloud and edge in their study collaboratively with a heterogeneous fog computing environment. In their proposed model, at first, they introduced a waited-cost model to find the most suitable configurations for incoming tasks from IoT applications. The model also utilises recurrent neural networks(RNN) to capture temporal behaviours across input features. After that, they use xDDRL in their prescheduling phase. And IMPALA for training their distributed brokers. The outcome of their study is to improve the execution time by 30%, energy consumption by 11% and weighted cost by 24%.

Zare et al. [44] proposed a novel approach to tackle the service placement problem (SPP) for IoT applications in fog computing environments. The authors introduce a deep reinforcement learning mechanism (DRL), which is the Asynchronous Advantage Actor-Critic(A3C) algorithm. Their model, A3C-based framework (A3C-SPP), is designed to minimise latency and cost while considering resource constraints. Their study treats SPP as a multi-objective optimisation problem and by utilises A3C to make dynamic placement decisions. Additionally, the A3C-SPP model includes a resource distribution extraction technique which aims to optimise resource utilisation over time. The simulation results demonstrate the effectiveness of A3C-SP against several other DRL approaches, including IMPALA and DDQL, in terms of improving several QoS parameters. In [45], Singh and Vidyarthi proposed a new solution for secure service placement in the fog-cloud ecosystem, combining machine learning (ML) and metaheuristic approaches. The proposed solution framework consists of two main components:

1. An ML-based classification, which is a modified adaptive Neuro-Fuzzy Inference System (ANFIS) for service classification.

2. A Hybrid metaheuristic algorithm that combines the Chaotic-based Grasshopper Optimization Algorithm (CGOA) and Genetic Algorithm (GA) to secure service placement.

The ANFIS model predicts suitable layers, either fog or cloud, for each incoming service, while CGOA-GA is responsible for optimizing service placement to enhance QoS, including security, cost, energy consumption, and makepan. Their approach is evaluated using Google cluster traces, and it shows incredible improvements compared to other methods in terms of makespan, computational cost, and energy dissipation. In paper [46], Abofathi et al. proposed a distributed learning automata-based approach to optimize the application placement model in fog computing. Their

proposed algorithm is a Distributed Learning Automata for module Placement (DLA-FMP). It is designed to optimize QoS parameters, including energy consumption and application loop delay, by mapping the modules to suitable fog nodes. The fog environment is modelled as a multi-layer architecture, with communication occurring between nodes on the same layer and those directly above or below to enable efficient model placement. Each fog node is assigned learning automata that cooperate to find optimal module placement solutions. The simulation result demonstrates the performance of DLA-FMP in energy utilization and latency reduction compared with other algorithms, including WOA-FMP, EGA-FMP, GAPSO-FMP and BPSO-FMP. On average, DLA-FMP achieves almost 16% improvements in energy consumption and an 18.21% reduction in application loop delay. Table 2.1 summarizes the application placement approach in fog computing by considering their case studies, proposed algorithms, performance evaluations, advantages and limitations.

| **Paper number** | **Case study** | **Proposed mechanism** | **Performance evaluation** | **Advantage** | **Limitation** |
|---|---|---|---|---|---|
| [28] | Service Placement in community network micro-clouds | BASP | Bandwidth, packet loss, response time | Improves bandwidth utilization, reduces pack loss, decreases client response time | Focus on service placement based on bandwidth and availability, but ignores other QoS factors like delay, which is essential in their scenario of "video streaming". |
| [29] | Healthcare | Module Mapping algorithm | Optimize network usage, application latency | Reduce delay, efficient utilization of network resources | Does not consider the dynamicity of mobility of IoT devices and end users. |
| [30] | Healthcare | Energy-aware allocation strategy | Network Usage, delay, energy consumption | Reduce delay, efficient utilization of network resources | Simple scenarios do not reflect the complexity of real-world healthcare scenarios,do not handle dynamic network changes. |

Table 2.1: Overview of Case Studies and Proposed Mechanism in Application Placement Approaches (Part 1)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [31] | General Case Study | Model Mapping algorithm | Optimizing network Usage, minimising Latency and Energy Consumption | Provides a structured approach to service placement, enhanced efficiency | Focus on static application characteristics, ignoring dynamic factors node failures and network changes. |
| [32] | Decentralized application placement | Combined of heuristic and ILP-based algorithm | Resource consumption, Average delay, Execution time and Efficiency | Effective for large problem instances, Scalable | High complexity, coordination overhead, simulation-based validation |
| [33] | Enhanced modular application placement | Heuristic algorithm LI-X | Network traffic, Total delay and Module distribution | reduce Response time, network traffic, and efficient resource utilisation | The study assume static resource availability and does not consider the impact of mobility. |
| [34] | Logistics centres in Industry 4.0 | DMGA | Cost minimisation | Cost-efficient deployment strategy and effectively optimise device placement | Moderate-scale scenario and lacks consideration of energy consumption and real-time dynamic changes. Focuses on static sensing in fixed locations, limiting the potential advantages of fog computing. |
| [35] | Data-intensive IoT application | Metaheuristic NSGA-II | Latency, cost and data availability | Reduce cost, latency and improve the performance of the system by increasing data availability | The Complexity associated with implementing the NSGA-II in large-scale fog computing environments. |
| [36] | Smart manufacturing industry 4.0 | Metaheuristic (MGAPSO and EGAPSO) | Service time, service cost, energy consumption | Reduce delay, cost and energy consumption | Did not consider interdependent IIoT applications, increases energy consumption with more fog nodes. |

Overview of Case Studies and Proposed Mechanism in Application Placement Approaches (Part 2)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [37] | General Case study | Hybrid metaheuristics GA-SA and GA-PSO | Latency, energy consumption and cost | Reduce makespan, energy consumption and cost-effectively | Does not consider task migration, assumes a static environment. |
| [38] | General Case study | Metaheuristic EMOPSOC and Fog Picker | Network latency, resource wastage, load imbalance efficiency | Reduce network latency by 80% minimize resource wastage , balance load | Assumes static environment |
| [39] | Smart City Applications | Integer Linear Programming (ILP) | Latency | Provide a flexible and efficient deployment strategy for IoT applications | Lacks comparison with other service placement solutions, Ignores Other QoS parameters like reliability and throughput, and is purely Theoretical with no experimental results |
| [40] | Predictive replica placement for a mobile users | Markov models(Fusion Multi Order Markov Model-FOMM) | Data availability, memory and network usage | Improves data availability and reduces global replication overheads | Require further testing to address dynamic changes in network conditions |
| [41] | General Case study | FONS | Network usage, energy consumption, delay, number of migrations | Support scalability and real-time processing needs of IoT applications | Needs further research to include dynamic characteristics and node failure scenarios |
| [42] | Smart City | MILP | Power consumption, data transmission and cost savings | Significant reduction in power consumption and data transmission | Ignore dynamic environment and assume only static environment. |
| [43] | Smart healthcare | X-DDRL | Improves execution cost, energy consumption, and system efficiency | Reduces exploration costs, improves sample efficiency | High exploration costs, limited adaptability to dynamic environment, no mobility models. |
| [44] | General Case study | A3C-SPP | Cost, Latency and resource utilization | Significantly reduce cost, latency and improve resource management | Requires high computational Resources, complex implementation. |

Overview of Case Studies and Proposed Mechanism in Application Placement Approaches (Part 3)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [45] | General fog-cloud ecosystem | ANFIS for classification CGOA-GA for service scheduling | Makespan, cost, energy consumption, and security | Significant improvement in makespan, computational cost, energy dissipation | High Computational complexity and parameter tuning; chaotic-GOA may have convergence issues. |
| [46] | General Case study | DLA-FMP | Energy consumption and application loop delay | Improve system performance by reducing application loop delay and energy consumption | Complexity increases with network size and requires parameter tuning. |

Overview of Case Studies and Proposed Mechanism in Application Placement Approaches (Part 4)

### 2.2.2 Task Scheduling

Task scheduling involves determining the optimal assignment of tasks to available fog nodes.

**Problem definition for task scheduling approach**

- **Objective:** To find the optimal path to assign tasks to fog nodes and optimise QoS parameters.
- **Assignment:** The solution involves the assignment of tasks $T = \{T_1, T_2, \ldots, T_n\}$ with various QoS requirements to available fog nodes $F = \{F_1, F_2, \ldots, F_M\}$ with different capabilities and locations.

**Task Scheduling in Practice:** Consider a smart factory with various sensors and devices. Resource scheduling strategies determine which fog nodes should process the data from these sensors and devices. All these processes should be monitored and controlled in real-time.

**Key challenges in resource scheduling approach:**

- Ensuring tasks are scheduled to minimise execution time and meet QoS requirements.
- Managing the dynamic nature of task arrivals and resource availability in fog nodes.
- Balancing load and avoiding bottlenecks.

According to the taxonomy depicted in Figure 2.2, task scheduling approaches can be classified into static, dynamic, hybrid and intelligent (AI-based) categories. Efficient task scheduling is paramount in fog computing environments to optimise the allocation of computing resources at the network edge. Unlike fog computing, where dynamic scheduling is often required, cloud environments typically rely on static resource scheduling strategies.

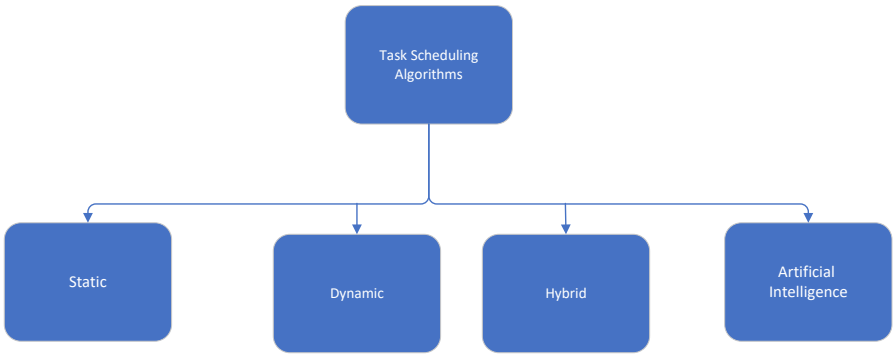


Figure 2.2: Hierarchical Representation of Task Scheduling Approaches

### Static Scheduling

In fog computing, static scheduling algorithms, such as FCFS (First Come First Serve), operate under the assumption that all information about tasks and fog resources is available in advance. This implies that the details of the task are known before the commencement of the scheduling process. The FCFS mechanism schedules resources based on a first-come, first-served basis, much like the First in First out (FIFO) system. In this mechanism, the first task that arrives in the fog devices is executed immediately, while subsequent tasks are stored in a waiting queue in ascending order based on their arrival. Similarly, SJF prioritises tasks based on their execution time, scheduling the shortest task first to minimise overall waiting time [47, 48].

Pham and Huh [49], focus on achieving a trade-off between execution time and cost in task scheduling for cloud-fog computing. Their heuristic algorithms prioritise tasks using an upward ranking method, ensuring that higher-priority tasks are assigned to resources first. The study aims to assign tasks to the most suitable nodes among the available nodes. Their simulation results demonstrate a balance between cost and execution time (makespan), showing that while the results are not always the most optimal, they are obtained efficiently. Another study conducted within the distributed scheduling category focused on QoS-aware scheduling for streaming

data applications in a fog computing environment. Their investigation demonstrates improvements in runtime scheduling while concurrently reducing delays and enhancing overall performance [50].

### Dynamic Scheduling

Dynamic scheduling is a contrast to static scheduling. Given the dynamic nature of edge devices and workloads, dynamic resource scheduling becomes pivotal. In this form of scheduling, task details are typically streamed in real-time, with no previous knowledge of the resource requirements. Dynamic scheduling calls for greater flexibility, especially when catering to applications that have unpredictable workloads at the edge. Ghanavati et al. [51] propose the DFTLA approach, a fault-tolerant task scheduling method that employs a variable learning automaton to actively monitor execution and make adaptive decisions based on environmental inputs. The goal is to ensure reliable task execution while optimising response time and energy consumption in fog computing platforms. Experiments validated its effectiveness, showing that it outperformed three baseline methods in terms of reliability, response time, and robustness while also achieving lower energy consumption. Bittencourt et al. [52] examine the scheduling challenges associated with movement applications in the fog and cloud context hierarchy. The approach improves response times and adapts to user mobility, but it requires sophisticated resource management to handle dynamic workloads and varying network conditions. Their study introduces different scheduling strategies to optimise resource utilisation while maintaining low latency; however, ensuring seamless application performance under unpredictable mobility patterns remains a challenge.

Yang et al.[53] propose a DEBTS algorithm for homogeneous fog networks to optimize energy efficiency and enhance system performance. Nonetheless, the approach requires careful tuning of control parameters to balance delay and energy consumption. Wan et al. introduce a unique job scheduling strategy called Energy-Aware Load Balancing and Scheduling (ELBS) [54] to optimize fog node resource utilization in smart factories. The scheduling strategy focuses on balancing energy consumption and delays by employing the Particle Swarm Optimization PSO algorithm.

### Hybrid

Hybrid scheduling combines elements of both static and dynamic scheduling to enhance flexibility and efficiency in task allocation. Unlike static approaches that depend on pre-defined task information, dynamic approaches adapt in real-time. Hybrid scheduling harnesses both methods to optimise resource utilisation. For instance, initial task assignments may follow a static plan, while adjustments occur

dynamically based on workload fluctuations. Methods such as the Stable Matching Algorithm (SMA) and meta-heuristic techniques like Particle Swarm Optimisation (PSO) exemplify this approach, ensuring a balance between predictability and adaptability in fog computing environments. Alfakeeh and Javed's paper [55] presents a task scheduling method based on a Stable Matching Algorithm (SMA) for fog computing integrated with IoT networks. It addresses the challenge of efficient and secure task allocation by leveraging the Gale–Shapley stable matching approach. The approach develops preference profiles for IoT tasks and fog nodes, taking into account aspects such as task delay and secrecy rate to ensure optimal scheduling. Simulation findings reveal that the proposed strategy decreases task delay by 33% and enhances secrecy rate by 200% compared to other existing algorithms, all while maintaining high resource utilisation ( 90%). Among task scheduling approaches, Time-Cost aware Scheduling (TCaS), is a recently proposed algorithm based on a genetic algorithm. Its primary purpose is to optimise task scheduling in cloud-fog computing networks. The algorithm was tested on 11 datasets. The test concluded that TCAS has far better optimisation than older algorithms, such as Bee Life Algorithm (BLA), Modified Particle Swarm Optimisation (MPSO), and Round Robin, with reductions of approximately 15%, 11% and 44%, respectively [56].

### Artificial Intelligence

Artificial Intelligence approaches can be classified as machine learning (ML) approaches or deep learning (DL) approaches. Currently, scholars investigate the feasibility of these approaches to enhance QoS. Recent studies on task scheduling in fog computing focus on the integration of ML and DL methodologies, particularly neural networks. These data-driven techniques leverage real-time monitoring, and adaptive decision-making to optimize resource allocation decisions at the network edge. Their approach helps to improve system efficiency by adjusting resources dynamically based on the workload demands. As a result, it enhances performance while minimizing latency and energy consumption [57].

Additionally, deep reinforcement learning models, such as Clipped Double Deep Q-Learning (CDDQL), provide an intelligent approach to resource scheduling in fog environments. The results of this study demonstrate the potential of these methodologies in improving energy efficiency, reducing service delay, and enhancing scheduling performance at the network periphery. They also highlight the advantages of these strategies in optimizing operational efficiency, though further refinements are needed for real-world applicability [58]. Another innovative resource management strategy, called Effective Resource Allocation Strategy (ERAS), is discussed. This strategy, tailored for healthcare applications in fog computing, employs a unique deep reinforcement learning method and fine-tunes its parameters using Particle

Swarm Optimization (PSO). The system aims to reduce latency and boost QoS indicators, including allocation expenses, response duration, bandwidth efficacy, and energy usage. ERAS outperforms preceding allocation approaches, achieving the least Makespan and the highest Average Resource Utilisation (ARU) and Load Balancing Level (LBL) [59]. Table 2.2 summarizes the task scheduling approach in fog computing by considering their case studies, proposed algorithms, performance evaluations, advantages and limitations.

| **Paper number** | **Case study** | **Proposed mechanism** | **Performance evaluation** | **Advantage** | **Limitation** |
|---|---|---|---|---|---|
| [47] | Healthcare scenario in fog computing | SJF | Response time, throughput, delay | Enhances job scheduling efficiency, reduces delay and improves system performance | Requires further validation in large-scale real-world healthcare environment |
| [48] | Task scheduling in fog computing | Multi-objective optimization using FCFS, SJF, GP and RR scheduling policies | Energy consumption, execution time, network usage, application delay | FCFS reduce energy consumption, execution time, and optimize average loop delay | Requires further validation in real-world environment and consideration of dynamic workload variations |
| [49] | Task scheduling in cloud-fog computing | Heuristic-based scheduling algorithm | Makespan and cost | Efficiently balances workflow execution time and cost | Study does not incorporate budget constraints or strict workflow deadline, which are suggested for future improvements |
| [52] | Scheduling application in fog computing considering user mobility | Mobility-Aware Scheduling algorithm | Network traffic and application response time | Effectively handles user mobility, ensures low-latency access | Requires advanced resource management to handle dynamic user requests and mitigate mobility prediction errors |
| [53] | Task scheduling in homogeneous fog networks | DEBTS | Energy consumption, average service delay | Balance trade-off between service delay and energy consumption | Needs further validation in real-world scenario include dynamic and heterogeneous fog environment |

Table 2.2: Overview of Case Studies and Proposed Mechanism in Task Scheduling Approaches (Part1)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [54] | Scheduling in smart factories | ELBS | Energy consumption, response time | Improve loading balancing and reduce energy consumption | Needs further validation in varied industrial environments to ensure scalability |
| [50] | DPS application in smart city | Distributed QoS-aware Scheduling for Storm | Performance , runtime adaptation and capabilities | Improves application performance | For complex topologies it may cause instability affecting application availability |
| [60] | Selection of path pairs in multicriteria ad hoc networks | GA | Performance in SNR and power consumption | GA provides stable performance and consistent path pair selection | Requires significant computation time and the results may vary under different conditions |
| [51] | Latency-sensitive and compute-intensive IoT applications | DFTLA | Response time, energy consumption | Reliable task execution optimized response time and energy consumption | Need more research on congestion handling in high-demands areas and inclusion of memory/task specific energy requirements |
| [59] | Resource allocation in healthcare application | ERAS using optimized reinforcement learning | Maximizes resource utilization, improves allocation cost, response time, bandwidth, efficiency, energy consumption | Efficient resource management and load balancing, enhanced decision-making in critical care | Requires further validation in authentic clinic setting |
| [57] | AI model simulation for resource management | Deep Q-Network (DQN) with GNN | Power consump-tion,efficiency, performance | Eco-friendly, energy-efficient, optimize power consumption | Need further research in security, privacy and real-world validation in diverse scenario |
| [58] | Task scheduling in fog computing for IoT application | CDDQL with parallel queuing | Service delay and average energy consumption | Provides energy-efficient task management | Requires further optimization for real-world deployment and scalability |
| [55] | Task scheduling in fog computing | SMA | Task delay, secrecy rate, resource utilization | Reduce delay by 33%, improve secrecy rate by 200% | Does not consider real-time workload fluctuations. |

Overview of Case Studies and Proposed Mechanism in Task Scheduling Approaches (Part2)

### 2.2.3 Task Offloading

The concept of task offloading is totally different from task scheduling. It occurs after a task has been assigned to a particular fog node or could happen before scheduling a task when the fog is not capable of executing the task due to several reasons, such as a lack of resources or the unavailability of the fog node. The task offloading concept can occur when tasks are offloaded to a single node or to multiple nodes. Offloading to multiple fog nodes is a complex decision because the task is distributed (split) into multiple fog nodes at the same time.

Overall, task offloading involves determining which task to offload from IoT devices to fog nodes or cloud servers.

**Problem definition for resource scheduling approach**

- **Objective:** To find the best strategies for offloading tasks, aiming to optimise and meet the requirements of QoS parameters.
- **Assignment:**The solution involves determining which tasks should be processed locally, task offloading to fog nodes, or sent directly to the cloud.

**Task Offloading in Practice:** Consider a smart healthcare system. Task offloading decides whether to process patient data on wearable devices, offload it to nearby available fog nodes for quick analysis, or offload it to a cloud server for complex processing.

**Key challenges in task offloading approach:**

- Offloading tasks to nearby fog nodes instead of distant cloud servers to reduce latency.
- Balancing the computational load across fog nodes and cloud servers.
- Minimising energy consumption and extending the battery life of IoT devices.

According to previous studies, task offloading can be classified based on the decision strategies, task status and the offloading extent, as shown in Taxonomy 2.3. Decision strategies can be either static or dynamic. It can be static when the offload of a task is predetermined, which happens based on expected task demands and fog nodes' capabilities. On the other hand, dynamic offloading happens in real time based on the current incoming task and the capability of the fog nodes. Task status can be critical when its urgent or requires a quick response, or it is important for system functionalities. Also, it can be non-critical when the latency is not sensitive to the task requirements. The extent of task offloading can be either full or partial. The full offloading approach occurs when all tasks from local devices are offloaded to fog nodes. Whereas partial offloading involves only selective tasks being offloaded to balance the load in the fog node.


Figure 2.3: Task Offloading Types

One standard classification of task offloading is based on the decision strategy, which includes static and dynamic offloading approaches. Several researchers propose a static approach to tackle task offloading in a fog computing environment. Some scholars introduce a static approach only, while others use both static and dynamic approaches in the same framework. The authors in the paper [61] propose a task offloading algorithm, the Gale-Shapley stable Marriage Matching Game (Stable Matching algorithm, SMA), in the context of 6G networks. Their approach addresses several QoS parameters, including energy consumption and latency. SMA matches IoT devices with fog nodes based on task size and energy availability. Tasks are offloaded to nearby fog nodes with sufficient resources to optimize QoS parameters. The simulation results show an improvement in energy consumption by 36% and a reduction in latency by around 37% , compared to systems without tasks offloading. In the paper [62], the authors propose a solution for task offloading and resource allocation issues by using GA-based and Gini coefficient-based algorithms. The authors formulate two scenarios:

1. Task caching during off-peak times: GA-based algorithm is used to optimize the caching of tasks on mobile user equipment (MUEs).

2. Task offloading during immediate needs: Gini coefficient-based algorithm is used for offloading tasks and allocating resources.

Caching helps their framework reduce computational burden and delay by caching popular tasks locally or sharing them via D2D communication. The proposed mechanism in this paper is a combination of static and dynamic approaches:

- Task Caching : This is a static approach because caching decisions occur during off-peak times and not continuously updated in real-time.
- Task Offloading and resource allocation: This uses a dynamic approach since these occur in real-time conditions. The simulation results show that the proposed algorithms improve system performance and QoS.

The authors in [63] propose a dynamic offloading model for flying fog computing in IoT networks. They leverage mobile drones equipped with fog nodes to improve IoT network performance by enhancing QoS, including delay and availability. Their model determines the optimal task assignment strategies, considering the mobility of drones, communication constraints, and computation capacity. The drones aim to process data at the network's edge, which helps to reduce the load from the central cloud servers and minimize the communication delay. Their simulations demonstrate the ability of their proposed solution to improve several factors, including latency, availability, and network performance, compared to the traditional static approach.

Tripathy et al.[64] propose the Dynafog framework to optimize task offloading in fog computing for IoT networks with multi-hop access points. Dynafog addresses task offloading issues and enhances QoS parameters, including delay and energy consumption. They formulate the task offloading problems as a non-linear optimization problem and convert it into ILP. This results in a non-convex problem that is complex to solve. The authors utilize a greedy heuristic approach in their framework to tackle the complexity and achieve efficient performance compared to other approaches. The framework takes into account multiple factors, including resource limitation in fog nodes and the dynamic characteristics of the environment. Dynafog reduces delay by almost 41%, 9% and 45% compared to Fog-MMKP, OptFogCloud and Robust-CompOff, respectively. Additionally, Dynafog reduces energy consumption by nearly 4%, 33% and 36% compared to Fog-MMKP, OptFogCloud and Robust-CompOff, respectively. study by [65] investigated dynamic, partial offloading tasks in fog computing. Their proposed algorithm, called Learning Repository Fog-cloud(LRFC) , was able to distribute tasks among available fog nodes efficiently. Their study aims to help delay-sensitive applications that need immediate response. Wei et al. [66] introduce a novel approach for task offloading in vehicular fog computing. It uses a many-to-many model; multiple vehicles serve as both service demanders and computation providers. Their proposed approach within their framework uses deep reinforcement learning, which is able to dynamically address vehicular needs to optimize QoS. Kishor and Chakarbarty, in their paper [67], proposed a

metaheuristic scheduler approach to tackle task offloading issues by using a smart ant colony optimisation algorithm (SACO). It dynamically selects the appropriate node to execute the incoming data from IoT sensors. The study compares SACO's performance against that of other schedulers like RR, MPSO, and BLA. Dang and Kim [68] proposed a dynamic approach named dynamic collaborative task offloading (DCTO) to address task offloading by considering the resource availability in fog devices. They used a parallel computing technique by dividing tasks into subtasks and executing them in one or two fog devices to reduce the delay of execution.

Another vital categorisation relies on task criticality, distinguishing between critical and non-critical tasks based on time sensitivity and system importance. The paper [69] introduces a new framework in fog-cloud computing for task offloading in enterprise management systems (EMS). The framework addresses two main challenges:

1. Critical tasks (delay-sensitive): These require immediate processing to ensure system performance.

2. Non-critical tasks: These are less time-sensitive but often energy-intensive

The authors propose a distributed task offloading algorithm based on noncooperative game theory. This approach adjusts task offloading dynamically based on the availability of cached services at the fog server. The critical task offloads nearby fog nodes when the services are available, while the non-critical tasks are processed locally or offloaded to the cloud if local resources are not available. A 0-1 knapsack method is also used for dynamic service caching, prioritizing task popularity to enhance system efficiency. The result of this investigation show a reduction of delay for critical tasks and a reduction in energy consumption for non-critical tasks compared to other existing algorithms.

The paper [70] introduces OffFog as a new framework to aid in defining offloading policies in fog computing environments. OffFog aims to optimize task and data offloading between fog and cloud layers. Their scenario considers IoT devices in smart cities with resource limitations. The proposed Framework provides systematic guidelines for making decisions and considers several factors, such as data compression, task status (Critical or Non-critical), and storage thresholds. Critical tasks are processed immediately via fog nodes, while non-critical tasks are offloaded to the cloud. OffFog demonstrates the ability to improve several parameters, including improving execution time by 5% and reducing latency by almost 76%. In the paper [71], the authors present a novel task offloading framework for large-scale asynchronous Mobile Edge Computing (MEC) systems. The proposed framework utilizes a Whittle Index (WI)-based policy for dynamic task offloading to maximize system performance while simultaneously handling the stochastic nature of task

arrivals and deadlines in an asynchronous MEC environment. All critical tasks are prioritized based on their urgency using the Shorter Slack Time less Remaining Workload (STLW) rule. STLW ensures critical tasks are completed before their deadlines. WI enables a scalable, low-complexity solution that balances the task offloading between energy consumption and task completion ratio. Simulation shows impressive results compared to other traditional algorithms. Paper [72] proposed an innovative solution for managing execution delay, payment cost of fog computing and energy consumption. Their proposed algorithm is a multiobjective optimisation framework aimed at optimizing offloading probabilities and transmitting power for mobile devices to fog nodes while balancing the trade-off between delay, cost and energy consumption. This framework also utilizes queuing theory to analyse the offloading process. Alasmari et al.[73] addressed task offloading issues through a new strategy called a multi-classifier-based algorithm (MCEETO). It is applied to select the appropriate fog node for a module placement by considering the node characteristics, Task requirements, network bandwidth and latency. This approach aims to enhance energy efficiency in fog computing environments.

Task offloading can also be categorised by the extent of offloading, which includes full offloading and partial offloading, depending on how much the task is delegated. Other researchers use either full or partial-task offloading approaches to improve QoS. Wang et al. [74] present a detailed analysis of full-task offloading in fog computing. They investigate the ability of fog computing to act as a remedy for delay-sensitive applications or whether it becomes a resource drain due to the frequent and massive offloading processes. The authors introduce the gravity model, which evaluates task offloading from the perspective of delay and resource consumption. Additionally, the model assesses the impact of large-scale task offloading on individual devices and the entire network. Through the proposed model, the authors find that when the number of fog nodes increases, the task lifetime and device effort decrease; however, the network effort does not scale efficiently. The outcome of their results captures several task offloading schemes. This study concludes by highlighting the benefits of full-task offloading in reducing individual task delays.

Chiti et al. [75] introduced a full-task offloading solution using matching theory in fog computing environments. Their proposed solution focuses on optimizing the offloading of task computation from IoT devices for Fog nodes. The main goal is to optimize service time while considering computational and communication costs. The authors consider the problem as a matching game with externalities, where each IoT device (task) selects the most suitable fog node based on several factors, including execution time, waiting time and communication delay. Additionally, their proposed solution utilizes a deferred acceptance algorithm to achieve an efficient allocation in a distributed manner and ensure stability during matching outcomes. The outcome of their investigation shows that their proposed solution can reduce the worst total

completion time, waiting time and mean total completion time per task compared to other traditional approaches.

On the other hand, A research study by [76] introduced an innovative approach for partial task offloading using the Gale Shapley stable marriage matching technique, which aims to reduce latency and enhance energy efficiency in fog environments for 5G networks. Their proposed solution matches the mobile devices and fog nodes to optimize QoS based on available energy and predicted requirements. Their simulation results show an improvement, with a decrease of around 30% in energy usage and latency. Fard et al.[77] proposed an improved non-dominated sorting crow search algorithm (INSCSA) for task offloading purposes to tackle several objective optimization challenges in fog computing. Their proposed model is able to optimize several parameters, including cost, energy consumption, and response time. The output of their investigation shows that INSCSA produces better results than several other traditional algorithms. Also, paper [78] proposed a new mechanism called fair task offloading (FTO), which aims to address task offloading issues. FTO has a similar target with MCEETO, which is selecting the appropriate fog node for incoming tasks; however, it is based on the history of energy consumption, priority, and capabilities of nodes.

Table 2.3 summarizes the Task Offloading approach in fog computing by considering their case studies, proposed algorithms, performance evaluations, advantages and limitations.

| **Paper number** | **Case study** | **Proposed mechanism** | **Performance evaluation** | **Advantage** | **Limitation** |
|---|---|---|---|---|---|
| [61] | IoT and 6G network | SMA | Energy consumption, and delay | Reduce latency, energy consumption, and improve the performance | Real-time constraints, scalability challenges. |
| [62] | D2D-aided fog computing | Coefficient-based task caching and Gini coefficient-based task offloading | Delay and energy utilization | Improve system performance, reduce delay and energy consumption | Complexity in real-time applications and large-scale scenarios. |
| [63] | IoT network with flying fog computing | Dynamic programming-based offloading strategy | Latency, availability, and resource utilization | Reduce latency, increase availability, and optimize resources | Complex in dynamic environment, drone mobility constraints. |

Table 2.3: Overview of Case Studies and Proposed Mechanism in Task Offloading Approaches (Part 1)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [64] | IoT networks with multi-hop access points | Dynamic task offloading with ILP and greedy heuristic | Delay , and energy consumption | Significant reduction in delay and energy consumption , and Improve system performance | Scalability with dynamic network topology remains to be explored. |
| [69] | Cloud-assisted fog computing for EMS | Distributed task offloading (game theory) and dynamic service caching (0-1 knapsack) | Task delay and energy consumption | Reduces delay and energy consumption effectively | Complexity in large-scale dynamic environment. |
| [70] | IoT devices in smart cities | OffFog for defining offloading policies | Network Latency, task execution time | Improve system performance | Further evaluation needed in more complex real-time environment. |
| [71] | Large-scale asynchronous MEC systems | WI-based offloading policy with STLW rule | Energy consumption, task completion ratio and total reward | Scalable, low complexity, and prioritize critical tasks | Complexity in practical implementation and real-time application. |
| [74] | Full task offloading delay in fog computing | Gravity model for task offloading and resource evaluation | Task lifetime, device effort, Network effort. | Reduce task delay, effective in describing offloading schemes | Potential resource drain in large networks, scalability issues. |
| [75] | Full task offloading in IoT-based fog computing | Matching game with externalities, deferred acceptance algorithm | Worst total completion time, mean waiting time | Reduce Worst total completion time, improve mean waiting time and total waiting time | Complexity in dynamic environments, assume fixed position of devices and nodes. |
| [65] | Service provisioning in fog computing for IoT application | Dynamic offloading-Using LRFC | Improve energy consumption and QoS | Enhances service provisioning by dynamically adapting to varying requirements | Need more investigation in how to handle large-scale deployments and heterogeneous environment |
| [76] | Task offloading in fog computing for 5G cellular networks | Dynamic offloading, Energy-efficient offloading algorithm | Energy consumption and latency | Improves energy efficiency and reduces latency | Requires further research to handle large-scale deployments and the complexity of real-world scenarios |

Overview of Case Studies and Proposed Mechanism in Task Offloading Approaches (Part 2)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [66] | Task offloading in vehicular fog computing for ITS | Dynamic offloading-Using MADRL | Dealy, System social welfare | Efficient resource utilization and higher long-term rewards for vehicles | Handling trading failures and decision-making latency is not covered. |
| [72] | Computation offloading in fog computing for mobile devices | Dynamic offloading-Using Multiobjective optimization algorithm | Energy consumption, execution delay, payment cost | Balances energy consumption, delay, and cost-effectively for mobile devices | Need more investigation in heterogeneous and large scale environments |
| [67] | Task offloading in fog computing for IoT application | Dynamic offloading-Using SACO | Task offloading time, delay | Improvs delay, enhances QoS by dynamically adapting to task requirements | Needs more investigation in heterogeneous and large-scale environments |
| [73] | Task offloading in fog computing for IoT application | Dynamic offloading-Using MCEETO | Energy, Network usage | Improvs energy efficiency, reduce network usage, enhances QoS parameters | Need more investigation in handling large-scale deployments and diverse IoT environments |
| [78] | Task offloading in fog computing for various IoT application | Dynamic offloading-using Fair Task offloading algorithm | Energy consumption, delay | Ensures fair energy consumption among fog nodes, reduce task delay | Needs further research for scalability and handling heterogeneous fog nodes |
| [68] | Task offloading in heterogeneous fog computing system | Dynamic offloading-Using dynamic collaborative task offloading algorithm | Average task execution delay | Effectively utilise resources and reduce task execution delay through dynamic task division and parallel computation | Requires further research to handle dynamic resource availability and large-scale deployment |
| [77] | Task offloading in fog computing for IoT application | Dynamic offloading-Using INSCSA | Response time, energy consumption, Cost | Efficiency balanced response time, energy consumption and cost, while considering system availability | Need more investigation in how to handle large-scale deployments and heterogeneous environment. |

Overview of Case Studies and Proposed Mechanism in Task Offloading Approaches (Part 3)

## 2.3 Conclusion

This chapter provided a qualitative comparative analysis of task-oriented resource management approaches in fog computing, specifically focusing on application placement, task scheduling, and task offloading. These approaches play a vital role in optimising QoS, improving system efficiency, and enhancing performance within fog computing environments. The main limitations of the qualitative analysis of task-oriented approaches are observed:

- **Lack of Multi-Objective Optimisation :** Most studies focus on optimising a single or limited set of performance metrics, such as delay or energy consumption, without considering a balanced trade-off among multiple conflicting QoS objectives like delay, cost, and power usage.
- **Task Scheduling:** A lack of adaptive scheduling mechanisms for heterogeneous fog devices.
- **Over-Provisioning and Under-Utilisation Issues:** Many approaches do not explicitly address the challenges of matching resource supply with application demands, leading to inefficient resource usage.
- **Limited Focus on Node Minimisation and Network Efficiency:** Existing solutions rarely consider strategies to minimise the number of active fog nodes while maintaining performance, which is critical for reducing operational cost and energy consumption.

These limitations highlight the need for further investigation into applying intelligent and efficient approaches to resource management in fog computing, as addressed in chapter 4 and 5 of this thesis.

# Chapter 3

# Advanced Resource Management

## 3.1 Resource-Oriented Management Approaches

This chapter emphasizes resource-oriented management, regarded as advanced due to its complexity in decision-making, such as dynamic allocation and workload balancing across nodes. Additionally, it introduces other aspects of resource management, including resource allocation, resource provisioning, and load balancing. The chapter also provides an overview of simulation tools for fog computing, accompanied by side-by-side comparisons.

### 3.1.1 Resource Allocation

Resource allocation refers to distributing a heterogeneous fog node $F = \{F_1, F_2, \ldots, F_M\}$ geographically to meet users requirements without violating QoS. Overall, the resource allocation approach is allocating appropriate resources for IoT services among available resources. This includes determining which resources (CPU, memory, bandwidth) should be assigned to different IoT services to ensure efficient performance and meet QoS requirements.

**Problem definition for resource allocation approach**

- **Objective:** to allocate resources to IoT services, aiming to optimise QoS parameters.
- **Assignment:** The solution involves assigning resources to services based on their requirements and the current availability of resources.

**Resource Allocation in Practice:** Consider a smart home system; the system assigns processing power and memory to various services such as security monitoring, climate control, and energy management. It does this based on their real-time needs.

The system also considers the availability of resources. **Key challenges in the resource allocation approach:**

- Matching resource availability with the demand of IoT services.
- Avoiding over-provisioning (wasting resources) and under-provisioning (failing to meet service requirements).
- Adapting to changes in resource availability and service demands.

Several researchers use various algorithms to tackle resource allocation issues, as shown in the taxonomy 3.1, which can be classified into either Heuristic-based techniques, Metaheuristic-based techniques, Auction-based techniques, or AI-based techniques.

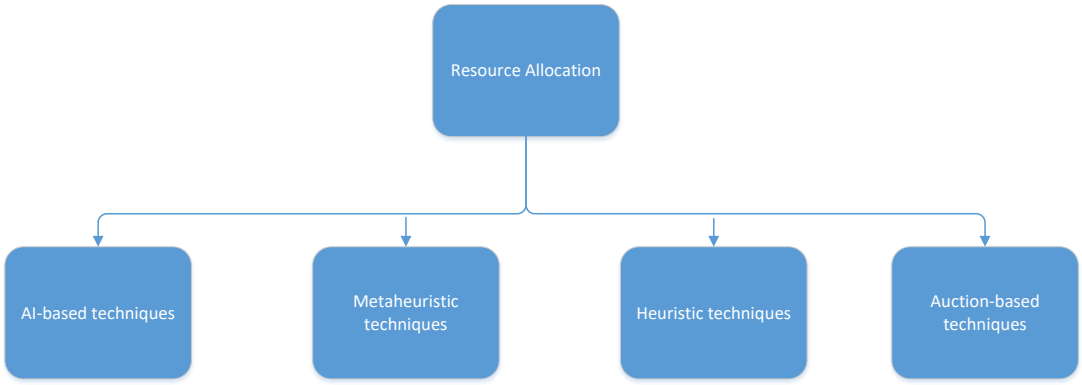


Figure 3.1: Hierarchical Representation of Resource Allocation Mechanisms

Heuristic approaches like Greedy, First-Fit, Best-Fit, and FCFS are generally less complex and aim to solve problems faster than other mechanisms. However, they may not always provide the optimal solutions.

Paper [79] proposed a scheduling algorithm called Path Clustering Heuristic (PCH) for resource allocation purposes. PCH's goal is to achieve a tradeoff between makespan and cost. It clusters tasks within a Directed Acyclic Graph (DAG) to reduce communication overhead and processing delays by grouping dependent tasks. This approach allows balanced allocation between cloud and fog resources. Their system model consists of three layers: of IoT, Fog and cloud architecture, with a centralized broker managing task scheduling and resource allocation. As a result, it enhances both execution efficiency and cost-effectiveness. In their simulation results, PCH showed better results compared to various other heuristic approaches, including

Heterogeneous Earliest Fish Time (HEFT), Critical Path on a Cluster Scheduling Heuristic (CCSH) and Cost Makespan Scheduling Strategy (CMaS) in terms of total time to complete all tasks (makespan) and cost of executing tasks. A study in [80] introduces a heuristic algorithm called PORA, which stands for predictive offloading and resource allocation. PORA is designed to optimize the computing location of tasks and maintain task offloading. It effectively reduces latency and enhances resource utilization dynamically. It is viable for real-time applications. PORA utilizes Lyapunov optimization to make adaptive decisions within a dynamic environment, balancing latency and energy efficiency. The authors investigate their approach with several mechanisms, including Random offloading, offloading to the nearest fog tier, no offloading, and offloading to the cloud.

Paper [81] addressed dynamic resource allocation issues by proposing a new approach to distributing resources based on the Petri nets, called priced timed Petri nets (PTPNS). It optimizes the allocation of resources, like storage resources and computing near users, by taking into consideration three factors: cost, task completion, and makespan. The PTPN model groups tasks with relationships (sequential or concurrent dependencies) to help predict cost and accurate time in a dynamic environment. Their simulation validates the approach's improvement in task completion efficiency over static methods. The remarkable thing about this approach is that the users independently select satisfactory resources from pre-allocated options based on these factors.

In the paper [82], the authors proposed a heuristic resource scheduling approach for fog-cloud computing environments. The case study focuses on general IoT applications and the proposed customized genetic algorithm to minimize deadline misses for IoT tasks by allocating resources in a three-tier-fog-cloud computing setup. GA is a well-known metaheuristic algorithm; However, the author refers to the use of GA in a holistic manner. They mean that they have customized and simplified the GA for their problem by finding a good solution rather than an optimal one. Their proposed solution, a customized GA approach, acts more like a problem-specific heuristic. The simulation results show their proposed solution can reduce deadline misses by 20% to 55% compared to other traditional approaches, including round-robin RN and priority scheduling PS.

In the paper [83], the authors proposed a novel approach to address the issue of energy-efficient resource allocation in device-to-device (D2D) assisted fog computing. It is called a low-complexity heuristic resource allocation strategy. The problem is formulated as a non-convex optimization problem, which is difficult to solve optimally. The authors' proposed solution introduces a two-fold approach:

1. The first method reformulates the original non-convex problem into a series of convex sub-problems, which can be solved by standard convex optimization techniques.

2. The second method introduces a heuristic algorithm for resource allocation to reduce computational complexity.

The simulation results show the proposed model reduces computational time and energy consumption. In the paper [84], Gai et al. introduce an Energy-Aware Fog Resource Optimization model (EFRO), a resource allocation technique for task allocation purposes. EFRO integrates standardization, smart shift operations, and a hill-climbing mechanism. The main objective of the heuristic algorithm is to generate near-optimal resource allocation solutions. The strategy is particularly effective for applications with time constraints. EFRO demonstrated impressive results in the simulation compared to other existing approaches, including MESF, RR, and DECM schemes, by saving energy consumption by up to 71.2%. Additionally, EFRO's generation time is much faster than other algorithms, making it a practical choice for dynamic fog computing. In paper [85] , Tan et al. propose a new resource allocation strategy to optimize QoS, specifically energy consumption and latency, in fog Radio Access Networks(F-RAN). The authors also consider fronthaul and latency constraints. First, the Authors formulated the resource allocation issue as a mixed-integer non-convex optimization problem, which is challenging to solve due to its complexity. To address this issue, they propose a heuristic iterative algorithm to convert non-convex problems into convex when binary decision variables are fixed. The algorithm optimally assigns computational tasks between available and suitable edge nodes and remote clouds. It aims to satisfy latency and fronthaul constraints while minimizing other QoS parameters. Their simulation results show that the proposed algorithm has better results in reducing energy consumption compared to other traditional techniques, including processing all tasks at the edge node or offloading them to the remote cloud.

Another study in paper [86] presents a new heuristic approach to tackle resource allocation issues in fog computing. Their approach dynamically allocates fog resources to the edge clusters based on the magnitude of data generated by these clusters. The proposed algorithm was evaluated in a distributed camera surveillance scenario on dynamic resource allocation based on data magnitude. The simulation results show the effectiveness of their approach compared to traditional cloud-only solutions in optimizing network utilization and reducing delay. Zhuang and Zhou, in paper [87] introduce a hyper-heuristic resource allocation (HHRA) mechanism to tackle resource allocation issues in fog computing. The objectives are to minimize system delay and energy consumption while meeting QoS requirements. HHRA consists of two stages:

1. High-Level Strategy (HLS): Ant colony optimization (ACO) is employed to select appropriate Low-Level Heuristic (LLH) in real-time. ACO is chosen for its dynamic adaptability and capability to optimize through pheromone-based feedback. This mechanism ensures ACO effectively avoids local optima and improves global solution accuracy.

2. Low-Level Heuristic (LLH): Operate on resource allocation solution space. The LLHs utilize global and local search strategies to enhance task scheduling efficiency and balance resource utilization.

Their proposed algorithm is able to dynamically adjust to changing fog computing conditions by continuously updating the pheromone matrix and adjusting the selection probabilities of the LLHs. The simulation results show that HHRA outperforms better than traditional algorithms, including GA and IPSO, in terms of optimizing QoS parameters.

Metaheuristic approaches, such as PSO, GA, ACO, etc., have been used instead of heuristic approaches to provide an optimal and efficient solution for resource allocation. Scholars who introduced these solutions considered several parameters to optimize, such as delay, network usage, execution time, waiting time, makespan, response time, throughput, cost, energy efficiency, and resource constraints.

Paper [88], Utilizes BLA algorithms to tackle resource allocation constraints in fog nodes to balance the tradeoff between memory requirements by users and CPU execution time. BLA efficiently utilizes the resource to assign Jobs to the most suited fog nodes. In this study, the job contains multiple tasks. BLA was tested against several other optimization algorithms, including PSO and GA, and it significantly optimized CPU execution time and memory usage by jobs. Paper [89] presented a study to allocate resources efficiently using an improved two Archive2 algorithm. It introduced a novel fitness evaluation method and a shift-based density estimation strategy (SDE). Their study constructed a four-objective optimization model to minimize service delay and cost while maximizing load balancing and stability in task execution. Their experimental results show that the proposed algorithm outperformed other approaches, like NSGA-III and MOEA/D, in improving resource allocation. Also, paper [90] proposed a hybrid meta-heuristic algorithm called Hybrid Mulit-Objective Crow Search Algorithm (CSA) for resource allocation and task scheduling in a fog computing environment. This approach simultaneously optimizes several parameters of QoS, including success ratio and security hit ratio, addressing the heterogeneous nature of fog devices. It was tested against several other algorithms like CSA and GA, across seven different scenarios. The proposed algorithm outperformed other approaches in terms of convergence speed, iteration count and solution efficiency, especially with a local search pivot rule.

The paper [91] presents a meta-heuristic framework for resource allocation and task scheduling based on Spider Monkey Optimization (SMO). SMO is a swarm intelligence algorithm inspired by the fission-fusion social structure of spider monkeys. Authors optimize the resource allocation using SMO to find the optimal fog node for processing tasks in fog environments. In addition, they propose several heuristic initialization methods, including LJFP, SJFP, and MCT, to initialize the population in SMO and find a good starting point for the optimization process. These heuristic approaches

help to improve the performance of SMO in finding optimal solutions. Their proposed model combines a mathematical optimization formulation with SMO to minimize time and monetary cost. The scenarios tested involve heterogeneous fog nodes with varying computational capacities and different tasks sizes to ensure diverse evaluation conditions. To summarize these approaches:

1. Meta-heuristic: The core algorithm is Spider Monkey Optimization SMO, a swarm intelligence-based meta-heuristic.

2. Heuristic: It incorporates heuristic initialization methods (LJFP, SJFP, and MCT) to improve the starting solutions for the meta-heuristic optimization process.

The results show that MCT-SMO outperforms other mechanisms, including PSO, in terms of minimizing cost and service time and improving overall performance.
Yakubu and Murali, in their paper [92], propose an effective two-stage resource allocation solution. The objective of their framework is to allocate resources across fog and cloud layers to handle delay-sensitive tasks. Their proposed model can be classified into two stages:

1. First stage: classify tasks based on a task guarantee ratio and categorize them to a suitable fog or cloud layer.

2. Second stage: Apply Bayes' classifier, which uses historical allocation data to allocate resources for new tasks effectively.

Additionally, a Crayfish Optimization algorithm (COA) is utilized to further resource allocation optimization. COA also balances task execution between fog and cloud layers to reduce delays and failures. Simulation shows the effectiveness of the proposed framework compared to traditional random, cloud-only and fog-only in optimizing several QoS parameters. Akintoye and Bagula, in their paper [93] propose a novel model for enhancing QoS through efficient resource allocation. The model includes two approaches:

1. Hungarian Algorithm-Based Binding Policy (HABBP): A heuristic method used to assign tasks(cloudlets) to virtual machines (VMs) optimally.

2. Genetic Algorithm-Based Virtual Machine Placement (GABVMP): A meta-heuristic approach designed to minimize costs and improve performance.

The simulation results demonstrate the effectiveness of both HABBP and GABVMP compared to other traditional methods for improving performance and QoS parameters.

Auction-based approaches are a mechanism of managing resource allocation via competitive bidding, where resources are allocated to the highest bidders. They aim to maximize resource utilization and can increase the revenue of resource providers by allowing dynamic pricing based on the current status of the market. This mechanism ensures an efficient matching between demand and supply. Gao et al. [94] introduce AVA, an auction-based VM resource allocation (AVA) in edge cloud nodes. AVA applies auction theory to handle competition for VM resources among mobile users while meeting deadline constraints. The problem is formulated as an n-to-one weighted bipartite graph matching problem with 0-1 knapsack constraints, which is proven to be NP-hard. To overcome this, authors develop a greedy approximation algorithm that ensures truthfulness, individual rationality, and computational efficiency. Extensive simulations on real datasets confirm the effectiveness of AVA.

Jiao et al.[95], present a decentralized auction mechanism for cloud-fog environments setup which is supporting public blockchains. In their study, blockchains were used for security purposes to manage bids, emphasizing the use of smart contracts to automate and secure auction processes. Their investigation concluded that a decentralized auction mechanism enhanced security in the allocation process and improved transparency, and ensured efficient resource allocation while maximizing social welfare.

Jain and Kumar [96] also introduce a double auction-based model on cost-efficient resource allocation in fog computing. Their proposed model focuses on to optimizing costs by enhancing economic properties, including truthfulness, budget balance and individual rationality. The use of smart contracts eliminates the need for a centralized auctioneer, enhancing security and fairness in resource trading. In paper [97], Luong et al. introduce a novel solution based on Auction theory for resource trading in fog computing. The solution focuses on blockchain applications. Their proposed solution is a combination of auction theory and deep learning, aiming to maximize the revenue for service providers. It also takes into account key economic properties, like incentive compatibility (IC) and individual rationality (IR). The proposed framework includes two neural networks for assignment and payment decisions. Both of which are trained using miners' bids for fog resources. The model optimizes the resource allocation within blockchain networks, which helps to reduce latency and improve overall system efficiency. Simulation results show that the machine learning-based auction outperforms traditional greedy algorithms by reducing IC and IR violations while increasing revenue. Jain and Kumar [98] propose a new solution based on an auction mechanism to tackle resource allocation issues. They utilize a blockchain-based fog computing platform to secure and optimize resource allocation. Their framework utilizes smart contracts between fog service providers and IoT devices to ensure secure transactions. The auction mechanism plays a vital role by allowing

IoT nodes to request bundles of resources, while fog nodes compete to provide these resources while considering prices, energy consumption, and delay. The descending combinatorial auction process is responsible for dynamically adjusting valuations to maximize fog node revenue and ensure fair resource distribution. Smart contracts manage the auction process to ensure security, transparency and fairness. Simulation results demonstrate that the proposed mechanism improves and enhances several factors, including profit generation for fog nodes, resource allocation and reduced delay compared to other traditional methods, including FCFS and fixed-pricing.

AI-based resource allocation utilizes advanced artificial intelligence technologies to enhance the allocation process. This process includes utilizing previous and current data to predict the future resource allocation process in real time dynamically. It aims to optimize system performance by adapting to sudden changes in requirements. Chen et al. [99] proposed a novel mechanism to minimise the perception-reaction time (PRT) in vehicular fog computing by integrating deep reinforcement learning for optimal resource allocation (computational and communication resources). Their approach was able to reduce PRT by more than 70% compared to the traditional methods. Moreover, this approach was able to enhance efficiency and road safety. Jamil et al. [100] introduce IRATS, a deep reinforcement learning approach using proximal policy optimization (PPO) to improve task scheduling and resource allocation in vehicular fog computing environments. IRATS classifies the incoming task based on priority and deadline constraints by considering link duration and vehicle mobility to minimise waiting time and task delay. The study also compares IRATS with some traditional strategies like DQN, A2C and random, which demonstrates that IRATS has better results in improving performance and reducing waiting time.

Lakhan et al. [101] investigate the ability of deep Q network-based resource allocation (DQBRA) to address resource allocation challenges in software-defined networks (SDN) in fog computing. They proposed a container-based architecture that utilizes DQBRA and considers some parameters like deadline, mobility and resource capacity. DQBRA was able to reduce energy consumption and application costs compared with some existing studies. Zhang et al., in their paper [102] propose a new technique to tackle resource allocation issues by introducing a deep reinforcement learning optimization algorithm in a blockchain-enabled fog computing system. Their framework aims to optimize the selection of fog nodes to allocate resources efficiently, task offloading strategies, and block sizes. The proposed algorithm utilizes the dueling deep Q-network (a DRL approach) to dynamically adapt the fog environment. It also ensures efficient resource allocation while maintaining system performance. The result, obtained through simulation, demonstrates improvements in system efficiency, energy consumption, and computation overhead compared to traditional Q-learning algorithms. Talaat, in paper [103], proposes the Effective Prediction and Resource

Allocation Methodology (EPRAM), a new mechanism to optimize resource allocation in real-time healthcare scenarios using fog computing. EPRAM consists of three modules:

1. Data Pre-processing Module (DPM).
2. Resource Allocation Module (RAM).
3. Effective Prediction Module (EPM).

The proposed mechanism manages resource allocation using Reinforcement Learning (RL) , and the predictions are performed with a Probabilistic Neural Network (PNN). This prediction handles healthcare data like heart attacks based on real-time data from IoT devices. The proposed framework aims to improve QoS, especially by reducing delay. The simulation results show that EPRAM produces better results compared to AWRR, WRR, LC, and RR.

Singh et al., in the paper [104] propose Collaborative Machine Learning (CML), a new mechanism to allocate resources efficiently designed for SDN-enable fog computing. CML allows decentralized training of models across fog nodes, which helps to improve resource management and QoS. Their framework uses a hybrid approach combining CML with Software Defined Networking (SDN) to optimize resource allocation. Simulation results demonstrate the effectiveness of CML in improving processing time, response time, and delay by 19.35%, 18.14% and 25%, respectively, while also reducing energy consumption by 7% and network usage by 9% compared to several traditional approaches.

Table 3.1 summarizes the Resource allocation approach in fog computing by considering their case studies, proposed algorithms, performance evaluations, advantages and limitations.

| **Paper number** | **Case study** | **Proposed mechanism** | **Performance evaluation** | **Advantage** | **Limitation** |
|---|---|---|---|---|---|
| [79] | General Case Study | PCH | Tradeoff between makespan and cost | Efficiently reduces execution time cost by clustering tasks in a cloud-fog setup | Assumes static resources availability; lacks handling for dynamic change in resource and workflow deadlines. |

Table 3.1: Overview of Case Studies and Proposed Mechanism in Resource Allocation Approaches (Part 1)

| **Paper number** | **Case study** | **Proposed mechanism** | **Performance evaluation** | **Advantage** | **Limitation** |
|---|---|---|---|---|---|
| [80] | General Case study | Heuristic-PORA using Lyapunov optimization | Power consumption reduction and queue stability, average latency reduction even with mild prediction error | Effective balances between latency and power consumption, adapts to varying system dynamics | Limited resilience to significant prediction errors; dependency on stable wireless channel assumptions. |
| [81] | General Case Study | Heuristic-PTPNs | Task completion time and cost | Improve resource utilization and QoS satisfaction | Lack consideration of dynamic resource demand and availability. |
| [82] | General case study | GA-based scheduling | Reduce deadline and improve performance | Efficient handling of time-sensitive IoT tasks | High complexity and computational cost. |
| [83] | D2D-assisted fog computing | Convex programming method and heuristic approach | Evaluated energy consumption, processing time, computational efficiency | Improve energy efficiency and reduce processing time | High runtime of convex method limits real-time use. |
| [84] | General case study | EFRO | Energy consumption Task allocation time | Significantly reduces energy consumption; faster task allocation | May encounter local optima due to the hill-climbing approach, sub-optimal in very large settings. |
| [85] | F-RAN with mobile users and remote cloud | Heuristic iterative algorithm | Energy consumption, latency, fronthaul constraints | Significantly reduces energy consumption, efficient resource allocation | Complexity, due to non-convex optimization, requires binary variable initialization. |
| [86] | General IoT application | Load-aware resource allocation heuristic | Resource utilization, network consumption, latency | Effectively reduce network consumption, delay and efficient resource allocation | Limited scenarios with fixed sensing rates, lacks node failure handling. |
| [87] | General case study | HHRA | Energy consumption, delay convergence | Improve energy efficiency and delay reduction | Slower initial convergence compared to IPSO, complexity increases with task size. |

Overview of Case Studies and Proposed Mechanism in Resource Allocation Approaches (Part 2)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [88] | Job scheduling for smart homes, cities and metering connected vehicles | Meta-heuristic - BLA | CPU execution and memory allocation | Optimal trade-off between CPU execution time and allocated memory, improve resource utilisation | Need more investigation in dynamic real-world scenarios |
| [89] | Resource allocation in fog-cloud computing for IoT application in the 5G era | Metaheuristic-Improved Two Archive2 Algorithm | Service delay, Cost, Load Balancing, task execution stability | Reduce service delay and cost | Scalability and adaptability in dynamic resources are not covered |
| [90] | Real-time secure resource allocation and scheduling in fog computing and IoT application | Metaheuristic-Multi-Objective CSA | Achieves better success ration and security hit ratios | Efficient resource allocation , improved success and security hit ratios | Several critical QoS parameters where ignored including delay, which is critical in healthcare domain and smart cities |
| [91] | General case study | SMO with heuristic initialization (LJFP, SJFP, MCT) | Service time , cost , monetary expenses | Reduces service time and cost | Increase complexity with larger task size. |
| [92] | IoT-cloud-Fog environments for general application | Task classification (Bayes's classifier and COA) | Execution time, latency and task failure rate | Reduce execution time and delay and improve the performance | Potential overhead due to Bayes's classifier and COA optimization, which may impact real-time task processing efficiency. |
| [93] | Cloud/Fog environments | HABBP for task allocation and GABVMP for VM placement | Allocation cost, processing time , resource utilization, energy efficiency | Reduces allocation cost, enhances QoS and improve resource management | Potential complexity in large-scale environment. |
| [94] | VM allocation for deadline sensitive tasks | Auction-based VM resource Allocation AVA mechanism with greedy approximation algorithm | Computational efficiency, successful ratio | Ensures truthfulness, individual rationality, computational efficiency, near optimal solution for VM resource allocation | Needs further research to handle dynamic changes in resource availability and tasks demand, greedy approach does not guarantee an optimal solution. |

Overview of Case Studies and Proposed Mechanism in Resource Allocation Approaches (Part 3)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [95] | Resource allocation for public blockchain networks using cloud/fog computing services | Auction-based Two bidding schemes (constant and multi demand) | Ensures truthfulness, individual rationality, computational efficiency | Maximizes social welfare, ensure truthfulness and individual rationality | Assume ideal communication, ignoring network congestion, latency, and bandwidth constraints. |
| [96] | Resource allocation in fog computing | Auction-based-Combinatorial double auction using blockchain and smart contract | Network utilization, pricing, and security analysis | Ensuring secure and fair resource allocation, improve network utilisation, prevents tampering through blockchain | Needs further research on resource and tasks demand. and do not consider user mobility. |
| [97] | Blockchain application in fog computing | Deep Learning-based auction mechanism | Revenue, IC, IR violation, service delays | Increases Revenue ensures fairness | Limited to a single resource unit, assuming a static environment. |
| [98] | Blockchain-based combinatorial auction for multi-task resource allocation in fog computing | Combinatorial auction with smart contracts | Fog node revenue, latency and resource utilization | Improve resource allocation, maximize fog node profit, and reduce latency | Complexity in managing combinatorial auction in large-scale systems. |
| [99] | Resource allocation in vehicular fog computing for ITS | DRL | Perception reaction time(PRT) | Significantly reduced PRT, Improve road safety and traffic efficiency | High computational complexity and reliance on stable V2V communication |
| [100] | Resource allocation and task scheduling in vehicular fog computing for ITS and IoV | AI-Based-PPO | Time task completion, packet loss, waiting time, end-to-end delay | Effectively prioritizes tasks, reduce waiting time, improve task completion rates | IRATS requires real-world validation due to its high computation cost, which might be limit deployment in real dynamic vehicular networks. |
| [101] | Resource allocation in SDN-enable fog computing for IoT applications | DQBRA for resource allocation | Cost, energy consumption, execution time | Increase efficiency reduces latency and energy consumption, and improve resource utilization | Does not consider data security in mobility network |

Overview of Case Studies and Proposed Mechanism in Resource Allocation Approaches (Part 4)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [102] | Blockchain-integrated fog computing framework | Dueling deep Q-network | Energy consumption and computation overhead | Reduce energy usage and computation overhead, improve system efficiency | Limited focus on scalability and security aspects, and not fully address real-world constraints such as communication delay |
| [103] | Smart Healthcare | EPRAM | Makespan, load balancing, average resource utilization, Latency, and energy consumption | Reduce latency and improve resource utilization | Scalability challenges due to high computational cost from deep RL and PNN integration. |
| [104] | SDN-enable fog computing | CML | Process time, response time, and network usage | Reduce execution time, power consumption and delay | Scalability challenges for large environment with more fog nodes. |

Overview of Case Studies and Proposed Mechanism in Resource Allocation Approaches (Part 5)

### 3.1.2 Load Balancing

Load balancing refers to the fog nodes when a particular node is overloaded or underloaded. It's a difficult issue due to their heterogeneity. Assume available m fog node $F = \{F_1, F_2, F_3, \ldots, F_n\}$, and we have Mp number of Application application requesting n Service $\{SV_1, SV2_2, SV_3, \ldots, SV_n\}$. If $F_2$ has a loading of computing less than other nodes or might be no load, it will establish load balancing issues. Overall, load balancing involves determining how to distribute workload evenly across available resources.

**Problem definition for resource scheduling approach**

- **Objective:** To find the best strategies for distributing workloads evenly across fog nodes, and meet the requirements of QoS parameters.
- **Assignment:** The solution involves determining which tasks should be processed locally, task offloading to fog nodes, or sent directly to the cloud.

**Load Balancing in Practice:** Consider a smart city traffic management system; load balance occurs when data processing tasks from various traffic cameras and sensors are distributed across several fog nodes. This ensures real-time traffic analysis

and response.

**Key challenges in task offloading approach:**

- Avoiding overloading of any single node while underutilizing other fog nodes.
- Maintaining a balance load to optimise resource utilisation and performance.
- Dynamically adjust the workload strategy based on resource availability.

Scholars investigated load balancing using several mechanisms. The taxonomy 3.2 gives a brief description of the techniques used to resolve load balancing, which are Fixed Distribution and Responsive distribution. Fixed Distribution, also known as static, refers to predetermined mechanisms for distributing the workload across all fog nodes without being affected by real-time changes. This mechanism relies on static data about the system and task requirements. It can be classified into either Deterministic or Probabilistic strategies.

- Deterministic strategies: It means allocation based on predefined rules considers node characteristics and task requirements.
- Probabilistic strategies: It means using the statistical method for task assigning, considering the static attributes of the system.

On the other hand, Responsive Distribution, also known as dynamic, refers to adjusting the workload in all fog nodes in real-time based on system performance, task requirements, and resource availability.It has four types of strategies, which are distributed, centralised, adaptive and non-adaptive strategies.

- Distributed: It means fog nodes are responsible for decisions based on partial system knowledge.
- Centralised: It means the controller is responsible for decisions based on complete system knowledge.
- Adaptive: This refers to modifying the policies of scheduling based on the current or past performance of the system.
- Non-adaptive: This refers to modifying the scheduling policies based only on the current performance of the system.

Singh et al. in [105] discuss algorithms like Round Robin (RR) and Weighted Round Robin (WRR) as simpler algorithms to implement in a fog computing setup to tackle load balancing issues. The RR technique is a basic method where tasks are cyclically distributed among fog nodes. However, it does not account for the real-time load or

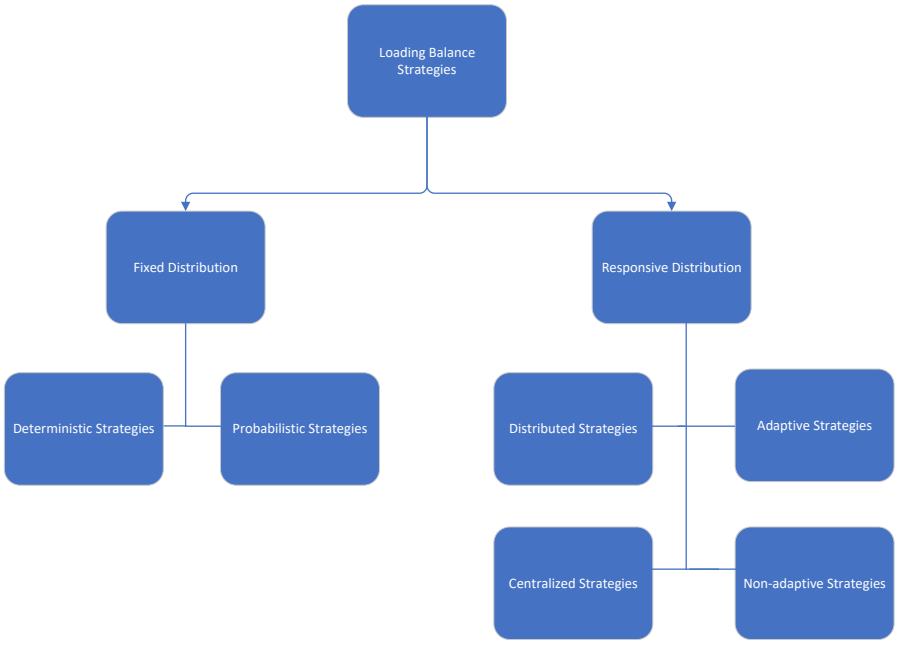


Figure 3.2: Strategies to Resolve Load Balancing

the capabilities of each node, making it less effective in more complex scenarios. The study emphasizes the benefits of static load balancing for predictable and uniform workloads. Alzeyadi and Farzaneh in [106] propose a static load-balancing technique in smart factories to optimize resource utilization. Their objectives focus on minimizing several parameters, including energy consumption, task waiting time, and communication delays by employing a dynamic threshold-based load-balancing algorithm. The proposed framework divides the scheduling and load-balancing into two categories: energy consumption and communication delay. The static approach monitors fog nodes and thresholds to prevent them from becoming either overloaded or underutilized. If a fog node exceeds the set threshold, tasks are redirected to other available nodes, aiming to optimize QoS parameters. Their simulation results show that the propose method reduces the communication network usage by almost 63% compared to ELBS algorithm. Additionally, it achieved better load distribution and improved the overall efficiency of the system.

Ali and Alubady in [107] introduce Remind Weighted Round Robin (RWRR), a new static load balancing approach to enhance resource distribution in fog computing environments. RWRR aims to ensure that tasks are equally distributed among fog nodes based on the available capacity of each fog nodes. The proposed approach is applied in a healthcare system scenario where real-time data processing is crucial for patient monitoring. RWRR is an improvement of the traditional WRR by

considering the capacity of each fog node and dynamically assigning tasks. The results of this study show that RWRR has improved system performance by almost 20% and reduce average response time near 120 ms. In the paper [108], the authors proposed a hybrid load balancing approach that includes both static and dynamic load- balancing methods. The authors proposed a Q-learning-based reinforcement learning (RL) algorithm to optimize task offloading and load balancing in the fog networks. Additionally, the system utilizes Software-Defined Network (SDN) to control fog nodes and improve the system's scalability. Their aim is to handle uncertainties, task demands, and fog node capacities while optimizing QoS, including delay and overload probabilities. Their hybrid approach divided into:

- Static Elements: Refer to the initial distribution of task among available nodes based on their capabilities.
- Dynamic Elements: Refer to Q-learning, which allows fog nodes to make real-time decisions on how to distribute the load based on incoming task requirements and the current capabilities of fog nodes.

The hybrid system improves load balancing and task distribution without requiring predefined assumptions about network topologies or workloads.

Paper [109] Proposed a reinforcement learning technique to tackle load balancing called ReTra. It dynamically adjusts the network state and distributes the workload across all fog nodes optimally. ReTa not only helps to manage the network traffic, but it also ensures the stability of the fog environment, even if the node suddenly fails, because the adjustments occur in real-time. Xu et al. [110] introduced a dynamic resource allocation mechanism (DRAM) to optimise load balancing in fog computing for IoT applications. DRAM integrates static resource allocation and dynamic service migration. It aims to tackle the load balance issue by allocating resources statically. Then, it dynamically adjusts allocation based on real-time resource utilization. DRAM outperforms better than several other approaches, including FF, BF, FFD and BFD, in resource utilization, the number of employed computing nodes and load balance variance. In Paper [111], the authors proposed a multi-level real-time scheduling algorithm (MLRTS) for load balancing in fog computing. MLRTS classifies incoming tasks into real-time or soft tasks. It also allocates resources dynamically and balances the load between fog nodes and the cloud. In their proposed mechanism, real-time tasks are prioritised, and resources are redistributed to maximize throughput and minimise execution time. Their simulation shows promising results for MLRTS compared with traditional mechanisms, including FCFS, Max-Min, PBATS, and RETS.

Wan et al. [112] proposed an energy-aware load balancing scheduling (ELBS) based on fog computing for smart factories. Researchers first used improved PSO

to establish an energy consumption model on fog nodes and then optimised load balancing. In their proposed solution, a multiagent system is utilized for distributed scheduling of manufacturing clusters. Their experiment results on a sweet packing line show that ELBS has improved several parameters, including load balancing, energy consumption, and operational performance. Boudieb et al. [113] proposed a novel deep reinforcement learning (DRL) algorithm to tackle load balancing and service selection. The proposed DRL algorithm is able to address the complexity of selecting appropriate micro-service instances dynamically and also meet QoS requirements while balancing the load.

The paper [114] presents DALBFog, a task-scheduling algorithm designed for deadline-aware and load-balanced task management in fog computing environments. Their proposed algorithm targets IoT applications and uses two sub-schedulers. These sub-schedulers map tasks to fog nodes based on their deadlines and current load conditions. DALBFog ensures efficient resource utilization. It meets the deadline requirements of delay-sensitive tasks. Their experiment results show that DALBFog minimizes average response time, execution time, and resource utilisation compared to other approaches like DRAM, FTLBSA, and MPGA.

Table 3.2 summarizes the load balancing approach in fog computing by considering their case studies, proposed algorithms, performance evaluations, advantages and limitations.

| **Paper number** | **Case study** | **Proposed mechanism** | **Performance evaluation** | **Advantage** | **Limitation** |
|---|---|---|---|---|---|
| [109] | Load balancing in fog computing for IoT applications | Dynamic load balancing- RL | resource utilization-Processing Delay-Network resilience | Balances workloads among fog nodes-enhances resource utilization-reduces delay and improve Network resilience | Need further research to handle large-scale deployments and diverse IoT application requirements |
| [110] | Load balancing in fog computing for IoT applications | Dynamic load balancing-DRAM | Resource utilization-Delay-Load balancing efficiency | Balances workload effectively among fog nodes-Reduce system delay-Enhances resource utilization | Need further research to handle large-scale deployments and diverse IoT application requirements |

Table 3.2: Overview of Case Studies and Proposed Mechanism in Load Balancing Approaches (part 1)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [111] | Load balancing in fog computing for real-tasks | Dynamic load balancing-MLRTS | Throughput-Resource utilization-Network utilization-Delay | Balances workload effectively among fog nodes-ensures timely execution of real-time tasks | Need further research to handle diverse application requirements and scalability issues |
| [112] | Load balancing and scheduling in smart factory environments | Dynamic load balancing -ELBS | Energy consumption-scheduling efficiency | Enhances energy efficiency-Balances workload, optimises scheduling in real-time | Need further research to handle large-scale and complex manufacturing environments |
| [113] | Load balancing and scheduling in fog computing for IoT applications | Dynamic load balancing-DRL | Reduces average failure rate by up to 65%, improves load balance by up to 45% | Optimizes service plan selection, enhances load balancing, reduces delay | Requires further research to handle dynamic fog environments complexity |
| [114] | Task scheduling and load balancing in fog computing for IoT applications | Dynamic load balancing-DALBFog | Improves resource utilization, reduces average response time, minimizes task execution time | Efficiently scheduling delay-sensitive task, balances workload, improves system performs | Needs further research to handle dynamic fog environments. |
| [106] | Smart factory resource management | Static energy-aware scheduling and load balancing | Energy consumption, task waiting time and communication delay | Reduce energy consumption and communication usage | Scalability for large systems may require further adjustment. |
| [107] | Load balancing in healthcare systems | RWRR | Response time, Resource utilization, throughput | Reduce response time, improve system performance | Scalability may be limited as system complexity increased. |
| [108] | Load balancing in fog networks | Q-learning-based RL with SDN control | Processing delay, overload probability and task offloading | Reduce delay and overload probability | Requires continuous network monitoring and training. |

Overview of Case Studies and Proposed Mechanism in Load Balancing Approaches (part 2)

### 3.1.3 Resource Provisioning

The Resource Provisioning (RP) concept refers to allocating and managing resources such as CPU power, storage, and bandwidth to meet the demands of application services. It involves setting up the necessary infrastructure, managing deployments

and operations, and maintaining resources. It could be critical to overprovision resources beyond the need for services because this will directly increase the cost and violate QoS. Similarly, underprovisioning the resources to less than the need for IoT services will violate service level agreements (SLA).
Overall, the resource provisioning approach is to scale resources up or down based on demand. This includes adjusting the number and the capacity of resources (Fog nodes, storage, processing power) based on the IoT services requirements.

**Problem definition for resource provisioning approach**

- **Objective:** To dynamically scale resources to match requirements, aiming to optimise QoS parameters.
- **Assignment:** The solution involves adding or removing resources based on real-time requirements and usage patterns.

**Resource Provisioning in Practice:** Consider a smart agriculture system. The system scales the number of fog nodes and processing power to handle varying data loads from sensors monitoring soil moisture, weather conditions, and crop health. This will ensure optimal performance during peak-demand periods and conserve energy during low-demand periods.

**Key challenges in the resource provisioning approach:**

- Ensuring sufficient resources are available without over-provisioning.
- Minimising cost and energy consumption associated with maintaining and scaling resources.
- Adapting to rapid changes in demand and resource availability.

This concept is different from resource allocation, which involves distributing available resources to various tasks or applications. RP focuses on making these resources available and optimally operational. Based on the literature review shown in Taxonomy 3.3, RP can be classified into Static, Dynamic or Hybrid resource provisioning approaches.
**Static approach:**
Occurs when resources are set up in a fixed configuration with no dynamic change based on current workload demands. This method is a predefined approach and is not suitable for real-time change but for environments with stable and predictable workloads.
**Dynamic approach:**
Occurs when resource configuration is allocated based on real-time demand. It's considered to be complex because it utilizes real-time data to adjust the configurations.

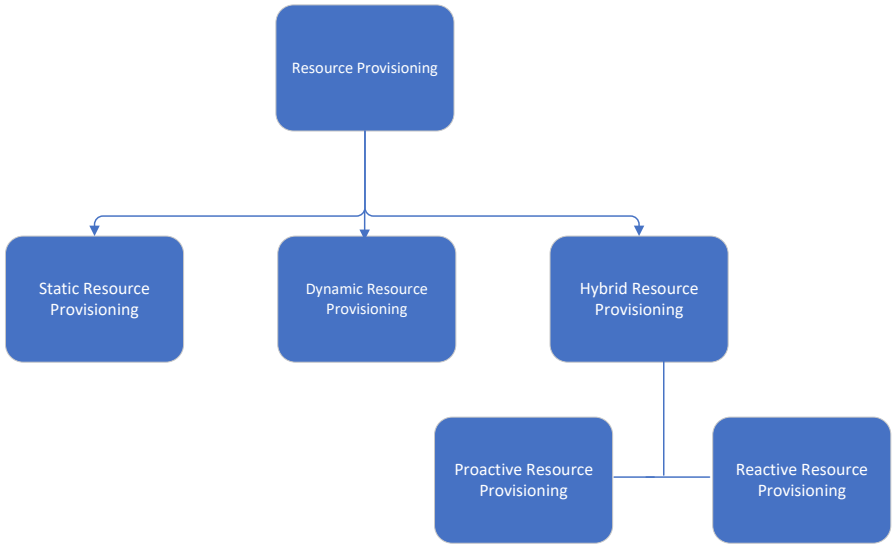


Figure 3.3: Types of Resource Provisioning

This approach can be either reactive or Proactive provisioning.
Reactive provisioning means allocating resources based on a workload response without prior knowledge or prediction-based planning.
Proactive provisioning means allocating resources in advance based on real-time analytics and predicting future demand using a historical data trends.
**Hybrid provisioning:**
It is a combination of static and dynamic provisioning strategies. A base level of configuration is statically set up, while dynamic adjustments occur based on real-time demands. This mechanism offer a balance between predictability and adaptability.

Few researchers apply the static provisioning approach, particularly in a private sector study involving smart offices and homes. This solution does not apply to workload change in a fog environment. Tasiopoulos et al. [115] proposed a static mechanism to tackle resource provisioning. The authors introduced a framework called Edge-MAP to facilitate resource distribution in bidding applications. They aim to address the resource provisioning challenges within large In-Network Computing Provider (INCP) infrastructure, especially with low latency applications. The framework focuses on resource provisioning for mobile users for tasks with restricted latency. Additionally, Vickrey-English-Dutch (VED) auctions utilized in the framework. The effectiveness of edge-map was evaluated in real-time vehicle traffic patterns with TAPASCologne dataset. Wang et al. [116] propose a novel framework, Edge Node Resource Management (ENORM). ENORM is designed to handle single and multiple edge nodes through provisioning, deployment, and

auto-scaling resources. It aims to reduce communication between cloud and edge nodes to optimize QoS, including reducing latency and bandwidth usage. ENORM demonstrates impressive results by reducing latency by 20-80% and decreasing data traffic to the cloud by 95%. Khalid et al. [117] propose a static resource provisioning and load balancing framework to optimize task distribution between fog and cloud layers in an IoT environment. Their proposed method aims to improve QoS through resource allocation techniques. The authors utilized K-means clustering and the Service-request prediction model (SRPM) to manage virtual machines. K-means and SRPM function to ensure that all inactive VMs go into a hibernation state to conserve energy. Additionally, their approach divides data into:

- Hot Data: Latency-sensitive data and migration needs, Allowing it to be processed by fog nodes.
- Cold Data: Bulk data, which will be sent to the cloud for long-term storage and analysis.

The experiment was tested in IfogSim, and the result show that the proposed model has improved several QoS parameters, including reducing latency, execution cost, and energy consumption compared to a cloud-only system.

The Authors in [118] introduce a novel static resource provisioning approach for smart buildings utilizing edge and fog computing technologies. The framework focuses on integrating different building subsystems, including energy management, climate control and security, by allowing IoT devices to process data near the source. The outcome of their investigations shows the ability of the proposed frameworks to reduce the bandwidth of communication between IoT devices and central data centers. Additionally, it distributes the tasks between edge and fog nodes, enabling real-time decisions, which reduce end-to-end latency. It is also noted that the proposed framework helps improve energy management in smart buildings by predefined static priorities without considering real-time workload changes.

In the dynamic approach, Etemadi et al. [119] demonstrate a learning-based mechanism for resource provisioning that dynamically manages the workload of IoT applications. Their proposed mechanism is a nonlinear autoregressive (NAR) neural network, which forecasts future demand and a hidden Markov model (HMM) to decide on scaling actions for resource allocation. The three contribution elements in this study are innovative decision-making via HMM, predictive resource management via NAR, and autonomic framework development to enhance resource provisioning. Another group of researchers in [120] aims to address the challenges of operational expenditure (OPEX) optimization in multi-access edge computing for systems with limited power supplies. Their proposed algorithm is a federated multidimensional fractional knapsack (FMFK). This investigation aims to handle resource provisioning

dynamically and predictively to balance energy cost and computational demand without violating latency constraints. The simulation shows that FMFK saves up to 40% compared with non-federated approaches. Paper [121] introduces an ElasticFog system to enhance resource provisioning in fog computing for IoT applications. Their approach utilizes Kubernetes to manage resource allocation dynamically based on real-time network traffic data. The outcome of their study is to reduce network latency and increase throughput.

Paper [122] proposes the FogSpot model, which represents a dynamic resource provisioning approach. It is a real-time pricing mechanism for allocating virtual machines (VM) based on current market demand and aims to enhance resource allocation efficiency. This process occurs through a third party, which is Cloudlests, to reduce latency in the application. Tseng et al. [123] proposed a new approach to address resource provisioning issues in fog computing by using Fuzzy-based real-time Autoscaling mechanisms (FRAS). They introduce a combination of integrated virtualisation (IV) fog platform and hypervisor technology with container virtualisation. Their aim is to develop virtual network functions for industrial applications. FRAS dynamically and efficiently adjusts the service scaling in fog nodes to optimise QoS, such as minimising delay and reducing operating expenses. Paper [124] presented a fog environment integrated with IoT to support real-time applications efficiently while considering energy constraints. Their approach is a heuristic aiming to minimise energy consumption without violating delay for IoT tasks.

In the hybrid approach, A study in [125] proposed an approach to tackle resource provisioning in smart cities. Their proposed model consists of several technologies, including Service function chaining (SFC) and various LPWAN. The authors also introduced a mixed integer linear programming model integrated with SFC and LPWAN to manage the constraints and optimize latency, cost and bandwidth utilization. Paper [126] introduced a Kubernetes-based management tool enhanced with self-adaption and network awareness to resolve container efficiency in a geo-distributed setting. They proposed a model-based reinforcement learning which will dynamically adjust how many containers are replicated and optimise their allocation via distributed locations. Usha and Rao [127] proposed a hybrid meta-heuristic approach to optimize resource provisioning in fog computing environments. Their proposed algorithm is ABC-JAVA, which is a combination of artificial Bee Colony (ABC) and JAVA algorithms to address the challenges in optimizing QoS, including reducing latency, improving throughput, and minimizing energy consumption in IoT-based fog computing. The strengths of the proposed ABC-JAVA come from both ABC and JAVA, allowing efficient global exploration and local refinement. The hybrid approach shows the ability to optimize multiple objectives, including maximizing throughput and resource utilization while minimizing energy consumption and latency. The simulation results show that ABC-JAVA outperforms other traditional

algorithms, including PSO, WOA, ABC, JAVA and EPSO. The improvement include increasing throughput by almost 89%, reducing latency near to 78 ms, and reducing energy consumption by 43%.

Also, Awotunde et al. in [128] proposed the HPSOFF-RPT, a combination of PSO and Firefly algorithm (FA), to enhance resource provisioning for IoT data fusion applications in cloud-fog environments. The HPSOFF-RPT model aims to optimize resource allocation by leveraging the strength of PSO and FA. PSO is used for global exploration, while FA ensures better local refinement. The HPSOFF-RPT model addresses several challenges in resource provisioning, including minimizing communication latency and optimizing resource utilization. The simulation results show that HPSOFF-RPT outperforms traditional algorithms, improving processing speed, reducing latency and enhancing resource allocation efficiency.

Table 3.3 summarizes the Resource Provisioning approach in fog computing by considering their case studies, proposed algorithms, performance evaluations, advantages and limitations.

| **Paper number** | **Case study** | **Proposed mechanism** | **Performance evaluation** | **Advantage** | **Limitation** |
|---|---|---|---|---|---|
| [119] | Resource Provisioning in fog computing for IoT application | Dynamic provisioning-Using Nonlinear Autoregressive neural network for prediction and HMM for decision making | Delay-Cost-energy consumption | Efficiently manages time-varying workloads and improve QoS | Need further research to handle large-scale deployments and heterogeneous environments |
| [120] | Resource Provisioning in multi-access edge computing | Dynamic provisioning-Using FMFK | Cost-energy consumption-e-OPEX | Enhances energy efficiency and reduce operational cost | Further research needed to handle dynamic and large-scale environments |
| [121] | Resource Provisioning in fog computing for IoT applications | Dynamic Provisioning-Using Elasticfog by Kubernetes platform | Throughput-Network latency | Enhances QoS by dynamically adjusting resource allocation based on real-time traffic | Need further research for large-scale and heterogeneous deployments |
| [115] | Edge resource provisioning for latency-sensitive applications | VED auction for resource allocation | Latency, resource utilization, system scalability | Latency, scalable, fosters competition among providers | Complex in implementation and maintaining fairness in large scale networks. |

Table 3.3: Overview of Case Studies and Proposed Mechanism in Resource Provisioning Approaches (Part 1)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [116] | Resource management for edge computing | ENORM | Application latency, data traffic, communication frequency | Reduces latency, data traffic and communication frequency | Study focus on static priority and ignore all dynamic priority. |
| [122] | Application Provisioning in edge/fog computing | Dynamic Provisioning-Using FogSpot spot pricing mechanism | Resource utilisation-Cost | Enhances QoS by efficient resource allocation through dynamic pricing | Need further research for large-scale and heterogeneous deployments. |
| [125] | Resource Provisioning in fog computing over low power wide area networks | Dynamic Provisioning-Using MILP | SFC-User latency-Data transfer time | Efficiently manages end-tend resource provisioning | Need further research for large-scale and heterogeneous deployments |
| [123] | Auto-scaling in fog computing for industrial applications | Dynamic Provisioning-Using FRAS | Average delay-Error rate-operating expenses | Dynamic, lightweight and low-cost auto-scaling solution, enhancing service scalability | Needs further research to handle more system metrics like energy consumption and user connections |
| [124] | Resource management in virtualized networked fog architectures for real-time IoT applications | Dynamic Provisioning-Using PABP heuristic | Energy consumption-Delay | Enhances energy efficiency and ensures real-time support for streaming applications | Needs further validation in diverse and large-scale deployment scenarios. |
| [126] | Deployment of containerized applications in geo-distributed fog computing environments | Dynamic provisioning-Using combines reinforcement learning and network-aware placement | Resource utilisation-Response time -Latency | Enhances scalability and reduces latency through network-aware placement and dynamic scaling. | Requires further validation for handling heterogeneous and large-scale deployments |
| [127] | Resource provisioning in IoT-based fog computing | ABC-JAVA hybrid algorithm | Throughput, energy consumption, and latency | Improve throughput, reduce energy consumption and lower latency | High computational complexity in large-scale systems. |
| [128] | Resource Provisioning in fog-cloud computing | HPSOFF-RPT | Latency, resource utilization and scalability | Reduce communication latency, improves resource allocation efficiency | Complexity in large-scale environments, requires fine-tuning of parameters. |

Overview of Case Studies and Proposed Mechanism in Resource Provisioning Approaches (Part 2)

| Paper number | Case study | Proposed mechanism | Performance evaluation | Advantage | Limitation |
|---|---|---|---|---|---|
| [117] | Resource provisioning in fog-cloud IoT environments | K-means and SRPM | Latency, energy consumption and execution time | Reduces latency, improving energy efficiency and enhances workload distribution | Complexity in large-scale deployments, requires constant monitoring of VM status. |
| [118] | Smart building service management | Static Resource Provisioning Approach using Edge and Fog nodes | Bandwidth | Reduces communication bandwidth, improve real-time processing | Limited adaptability to dynamic workload changes. |

Overview of Case Studies and Proposed Mechanism in Resource Provisioning Approaches (Part 3)

## 3.2 Simulations in Fog Computing: Advantages

This section discusses Simulation tools utilized for resource management to optimize QoS in fog computing environments. According to the literature review, scholars use simulation tools more often than real-life systems. One study shows that more than 90% of task scheduling studies were implemented in simulations, with only 10% in actual fog environments [129].

Most scholars prefer simulation tools over real-life systems for several reasons, including cost-efficiency, scalability, flexibility, reproducibility, time efficiency, safety and risk management, and global accessibility aspects. Here are the elaborations for each aspect.

- **Cost-Efficiency Aspect:** This is a prime reason for using simulations rather than real-life systems, as they help to avoid the expense of devices such as IoT nodes, edge servers, and networking equipment. Simulations allow researchers to apply their own scenarios without investing in costly physical resources.

- **Scalability Aspect:** Another reason for using simulations is their scalability, which allows researchers to test large-scale scenarios with even millions of devices. These tremendous scenarios would be impractical or impossible to deploy in real life, making them useful to test the effectiveness of various approaches. Also, simulations allow researchers to explore future scenarios of fog node environments with more devices, networks and services than currently exist, enabling forward-looking study.

- **Flexibility Aspect:** In simulation tools, it is easy to adjust parameters like network latency, device performance, and workload distribution without

being limited by hardware. Additionally, simulations allow researchers to test customized scenarios with custom configurations, protocols, and algorithms in a way that might not be feasible in real-life environments. They also allow the testing of extreme conditions such as extremely high traffic, failure conditions and rare security breaches, which would be hard to safely and accurately create in the real world.

- **Reproducibility Aspect:** Simulations ensure consistent results, whereas real-life systems can behave unpredictably due to external factors such as hardware wear, weather, or network congestion. Simulations remove these inconsistencies, allowing researchers to replicate experiments with identical conditions. Moreover, researchers can share their simulation models and configurations with other communities to replicate and validate results under the same environment, allowing comparison across other studies.

- **Time efficiency Aspect:** Time-saving is a key objective of using simulations, allowing researchers to observe long-term system behavior quickly. Researchers can rapidly iterate on new algorithms, protocols and architectures without consuming time to resetting physical resources.

- **Safety and risk management Aspect:** Simulations allow for safe testing of failures in a risk-free environment, whereas in real life one issue like a network attack experiment, energy depletion or node failures can be dangerous and costly. Simulating security attacks and defenses in a virtual environment avoids the risk of damaging real infrastructures.

- **Global Accessibility Aspect:** Researchers from various institutions that do not have access to advanced hardware can still investigate state-of-the-art fog and edge computing concepts by using open-source or cloud-based simulators. Simulations can also be exchanged among various teams and institutions, fostering collaboration without the necessity of physical device configurations.

- **Testing of Emerging Technology Aspect:** Simulations allow researchers to investigate and test different novel architectures, scheduling algorithms, and resource management strategies before real-world deployments. This feature helps researchers identify inefficient technologies and enables industries to avoid them.

### 3.2.1 Overview of Key Simulation Tools in Fog Computing

This subsection will discuss the most well-known simulators commonly used by researchers in implementing fog computing and applying their resource management

approaches.
One popular simulation tool that can be easily observed in most academic papers is IfogSim [130], which is designed for a fog computing environments. [131] shows that 36% of the literature uses IfogSim as the simulation tool to evaluate their proposed resource management approaches. Scholars initially introduced CloudSim [132] as a simulation framework that provides an interface for seamless simulation, modelling, and experimentation of cloud computing hypotheses. It offers a platform for testing newly developed application services in a controlled setup environment, helping developers test the service performance of applications. However, CloudSim cannot handle the IoT layer, so the IfogSim toolkit was developed on top of it. Statistically, there are 22 packages belonging to CloudSim, with only 13 packages being different from CloudSim in IfogSim. IfogSim is a toolkit based on Java that can simulate and model fog computing environments and assess scheduling policies upon fog and cloud resources. Here are the main features that can be found in This tool:

- **Open-source simulation environment.**
- **Support simulation and modelling of large-scale cloud computing and fog computing environments.**
- **Basic network simulation.**
  Simulates communication between IoT devices, fog nodes, and cloud, through lacks detailed network protocol simulation.
- **Supports containerized services.**
  Virtualization engine for the creation and management of virtualized services(e.g., VMs, containers).
- **Customizable resource management policies.**
  Supports custom resource scheduling and task allocation.
- **Support for IoT environments and devices.**
- **Custom topology creation.**
  Allows researchers to define complex topologies for for and cloud environments.

These are the main drawbacks of IfogSim:

- **Limited Networking Feature.**
  IfogSim focuses more on fog computing and IoT systems but lacks detailed network protocol simulation compared to tools like FogNetSim++.
- **Complex Setup.**
  Researchers need to be familiar with Java and have some technical expertise to set it up.

- **Limited Mobility Support.**
  While it supports mobility, it is not as advanced as tools specifically designed for mobile fog environments like LEAF.

Another simulation is EdgeCloudSim [133], based on Java, and it supports all platforms; however, it does not support all resource management approaches like scheduling algorithms.It is highly effective in simulating the interaction between edge devices and cloud services, and also provides a detailed performance analysis for workload distribution between these layers. It emphasizes the integration of cloud and edge computing, allowing users to model communication between cloud and fog layers. The clear drawback of this simulation is that it lacks the in-depth networking features found in other simulations.
These are the main features of EdgeCloudSim:

- **Workload distribution.**
  Simulates and analyzes the performance of different workloads distributed between cloud and edge devices.
- **Cloud-edge communication modeling.**
  Enables simulations of communication between edge devices and cloud resources.
- **Latency and response time analysis.**
  Provides insights into latency and service response times in edge computing scenarios.

These are the main drawbacks of EdgeCloudSim:

- **Lack Advanced Scheduling.**
  EdgeCloudSim does not fully support complex resource management and scheduling algorithms.
- **Basic Networking Capabilities.**
  The tool lacks the ability to model detailed network protocols and traffic in comparison to other simulators like FogNetSim++.
- **No Energy Consumption Modeling.**
  It does not offer built-in modules for evaluating the energy consumption of fog or edge nodes like IfogSim.

Cooja [134], a simulation based on C and Java languages, supports multiple platforms, including Linux, Windows, and macOS. It focuses on wireless sensor networks (WSNs) using the Contiki operating system. It is well-known for utilizing low-power and lossy networks (LLNs) in IoT environments, such as simulating the behavior of

sensors, actuators and network communication in IoT systems. Although Cooja does not directly simulate fog computing, researchers still integrate it to model network protocols and test small-scale systems for energy efficiency.
YAFS (Yet Another Fog Simulator) [135] provides flexibility in simulating large-scale distributed computing environments with complex communication networks. It is a suitable tool for event-driven simulations, such as when tasks are offloaded and processed by fog nodes dynamically based on network conditions. It is an ideal choice for studies focusing on latency and service response times in highly distributed entremets.

These are the main features of YAFS:

- **Event-driven simulation.**
  It can model the dynamic behaviour of fog nodes as they respond to task offloading based on network conditions.
- **Customizable network topologies.**
  Allows researchers to define complex, large-scale distributed systems and simulate network performance.
- **Latency and bandwidth evaluation**
  Supports detailed analysis of network latency and bandwidth usage for fog nodes.

These are the main drawbacks of YAFS:

- **No graphical Interface.**
  A lack of GUI, which makes it harder compared to other simulations to set up and run your simulations without deep technical expertise in the tool.
- **High Learning Curve.**
  Requires significant customization; researchers need to be familiar with Python to configure it effectively.
- **Limited Energy Simulation.**
  It Does not consider energy efficiency or power consumption modelling, which are highly important in fog computing research.

FogNetSim++ [136], a simulation built on top of OMNeT++ [137], focuses on modeling communication protocols, network traffic, resource allocation, and data flow between devices, fog nodes, and the cloud. Here are the key features of the simulator:

- **Detailed network protocol modeling:**
  Built on OMNeT++, offering advanced simulation of network communication protocols.

- **Network Traffic Modeling:**
  simulates network traffic between IoT devices, fog nodes and the cloud.
- **Fog and IoT Communication:**
  Models communication protocols and data flow in hierarchical fog computing architectures.
- **Resource Allocation:**
  Can analyze task offloading and load balancing between fog nodes.
- **Energy consumption:**
  can evaluate energy efficiency in distributed systems involving fog nodes.

The drawback is that it focuses on network-level aspects, not as comprehensive in modelling high-level fog computing scenarios, such as application placement, like IfogSim. However, it remains highly useful for researchers investigating the communication and network aspects of fog computing. These are the main drawbacks of FogNetSim++:

- **Focus on Network layer:**
  It is not comprehensive for application-level modelling, like task placement or resource scheduling.
- **Requires OMNeT++ Expertise:**
  It inherits the complexity of OMNeT++, which requires expertise to set up and run detailed simulations.
- **Limited Application-level modelling :**
  Focuses More on the network aspect and is not suitable for high-level resource management and application placement studies.

A newer addition to the list of these simulators is LEAF (Large Energy-Aware Fog computing) [138]. LEAF provides high-level simulations of fog, edge, and cloud environments with a strong focus on energy consumption. It is a useful tool for implementing energy-aware algorithms that dynamically adjust task allocation to minimize power usage in a distributed environment. This makes LEAF particularly useful for applications like smart cities, smart grids and other energy-sensitive IoT systems. It is designed for energy-aware simulations and enables dynamic modelling of mobile nodes, power consumption and energy-saving algorithms. LEAF utilize the SimPy library for process-based discrete-event simulation and NetworkX for modeling infrastructure graphs.These are the main features of LEAF:

- **Energy-aware task scheduling:**
  Can simulate task allocation based on energy consumption to minimize power usage.

- **Dynamic infrastructure modelling:**
  Can model mobile nodes and dynamically change fog and IoT environments.
- **Energy consumption analysis:**
  Provides full analysis of power usage patterns in fog, edge, and cloud environments.

These are the main drawbacks of LEAF:

- **Energy-Focus:**
  Simulator has a built-in Energy model but offers limited evaluation of other factors, such as detailed networking, including delay and bandwidth calculations.
- **Moderate Scalability:**
  It may not be as scalable as other simulators like IfogSim when dealing with very large systems.
- **Limited Focus on Fog-Specific Scenarios:**
  LEAF focuses on energy consumption, which may not cover the full spectrum of fog computing scenarios like resource management and task offloading.

Independent Python and MATLAB implementations: refer to scholars who implemented their own architecture using Python libraries like Flask, Celery, or Dask. In these implementations, researchers have full control over the architecture and can tailor the system to specific fog computing requirements. However, these approaches require significant development time and effort compared to utilizing toolkits like iFogSim or LEAF.
These are the main features of Custom fog computing environment:

- **Flexible architecture:** Full customization to meet specific fog computing needs.
- **Custom task scheduling:** Use libraries like Flask, Dask, or Celery for tailored scheduling and resource management.
- **Integrated with ML:**Support machine learning and data analysis via Python and MATLAB libraries.

These are the main drawbacks of a custom fog computing environment:

- **Time-intensive:** Requires significant development effort compared to pre-built tools.
- **No built-in models:** Lacks ready-to-use features like energy modeling or QoS metrics.

- **Limited support:** Less community and documentation support than established simulators.

Table 3.4 provides side-by-side comparisons for each simulation. Each tool provides different strengths based on the system size, complexity, and specific parameters of interest. Deep investigations show that IfogSim and LEAF are the best options for optimizing QoS via resource management approaches. However, choosing suitable simulators depends on the problems the researchers aim to solve.

| Tool Name | Programming Language | Platform | Network Configuration | Scheduling Algorithm | Energy Module | Fog computing | IoT | Scalability | Support for Mobility |
|---|---|---|---|---|---|---|---|---|---|
| Cooja [134] | C & java | Linux,Windows, macOS | ✗ | ✗ | ✗ | ✗ | ✓ | Limited | ✗ |
| EdgeCloudSim [133] | Java | ALL | ✗ | ✗ | ✗ | ✓ | ✓ | High | ✗ |
| Fogtorch [139] | Java | ALL | ✗ | ✗ | ✗ | ✓ | ✓ | Limited | ✗ |
| CloudSim [132, 140] | Java | ALL | ✓ | ✓ | ✓ | ✗ | ✗ | High | ✗ |
| IfogSim[130] | Java | ALL | ✗ | ✓ | ✗ | ✓ | ✓ | High | ✓ |
| YAFS [135] | Python | ALL | ✓ | ✓ | ✗ | ✓ | ✓ | High | ✓ |
| FogNetSim++ [136] | C++, OMNeT++ | ALL | ✓ | ✓ | ✓ | ✓ | ✓ | High | ✗ |
| LEAF [138] | Python | ALL | ✗ | ✓ | ✓ | ✓ | ✓ | Moderate | ✓ |
| Independent Python implementation | Python | ALL | ✓ | ✓ | ✗ | ✓ | ✓ | Varies | ✓ |

Table 3.4: Common Simulators for Fog Computing Environments

## 3.3 Conclusion

This chapter continues to present a comprehensive literature review on advanced resource management strategies in fog computing environments, focusing on optimizing QoS. The chapter begins by highlighting the advanced resource-oriented management aspects, which are resource allocation, load balancing and resource provisioning. Additionally, it discusses simulation tools for fog computing environments with detailed comparisons.

Based on Chapters 2 and 3, resource management is classified into six key categories: application placement, task scheduling, resource allocation, task offloading, load balancing, and resource provisioning. During the review process, it became apparent that some researchers introduced some of their work under certain category titles, but their actual investigations belonged to different categories. This occurred for two reasons:

- They grouped task scheduling, resource allocation, and task offloading into the same category.

- They consider task offloading and load balancing as the same category.

However, there are distinct categories, as demonstrated by different comprehensive surveys and workshops. These few papers with misleading titles were reclassified based on the actual focus of their work. Additionally, several papers addressed multiple issues and used objectives that spanned more than one category. For example, some papers tackle task scheduling and load balancing or application placement and resource allocation simultaneously [38, 79, 80, 83, 89, 90, 91]. For clarity, these papers are classified based on the primary category to which they contributed the most. It is also notable that several papers employed different approaches in their proposed solutions, combining Meta-heuristics with AI or combining heuristic and meta-heuristics, to address and distribute their model in different mechanisms as in papers [32, 38, 87, 91]. Each category begins by outlining the challenges researchers may encounter during their investigations. To enhance clarity, several example scenarios are introduced for each category. Additionally, the problem definitions are introduced with the classification of previous research approaches that address these challenges. These classifications not only offer current methodology and structure understanding of various approaches but also help future investigations aiming to enhance QoS in fog computing.

Every paper has its own limitations, which are listed in the side-by-side overview summary for each category. Here are the main challenges across all previous studies:

- **Unrealistic Simulation Environments:** Many studies utilise a large number of fog nodes in their simulations without considering the cost of these nodes and infrastructure constraints, which do not reflect real-life scenarios.
- **Lack of Intelligent Resource Allocation Approaches:** The majority of existing works rely on conventional scheduling, heuristic, metaheuristic, or swarm intelligence methods, with limited adoption of advanced neural models such as graph neural networks or deep learning for dynamic resource scheduling.
- **Resource allocation:** Inefficient dynamic resource allocation strategies for QoS assurance.
- **Latency Optimisation:** Current approaches may not effectively address end-to-end latency across various use cases.
- **Limited implementation:** Current studies often target a specific domain, such as smart healthcare, with a lack of generalisation to other fields like smart agriculture or smart cities. This limits the applicability of the proposed solution across diverse real-world scenarios.

This chapter aims to motivate researchers and industry to explore unresolved challenges in and innovate solutions for how to optimise QoS, which will help improve the reliability, scalability and performance of fog computing environments.

# Chapter 4

# Enhancing QoS using GGCN-Based Resource Allocation in Fog Computing Environment

## 4.1 Motivation

Resource allocation is a key challenge in distributed systems, especially with fog computing. This inefficiency impacts critical QoS metrics, including latency, network usage, cost, reliability, and response time. Addressing these challenges improves operational efficiency and enhances the end-user experience by optimizing service delivery. To this end, Gated graph Convolutional neural networks (GGCNs) offer a promising solution by leveraging graph structures to handle complex resource allocation problems in fog environments.

GGCNs are an advanced mechanism of graph neural networks (GNNs). Researchers designed them to leverage the strength of graph convolutional networks (GCNs), which use gating mechanisms to improve information propagation through graph structures. They are employed to handle complex graph-structured data, offering solutions to improve performance in various domains, including computer vision, natural language processing, and wireless communications.

Traditional GCN algorithms excel at aggregating information from local neighbourhoods but usually struggle to capture long-term dependencies due to over-smoothing. GGCNs, on the other hand, combine local feature extraction with long-

Portions of this chapter are sourced from my published paper: *Delay and Total Network Usage Optimisation Using GGCN in Fog Computing*[141].

range dependency capture. By integrating the gating mechanism, GGCN addresses this issue, drawing inspiration from gated recurrent units (GRUs). It passes relevant information while filtering out the noise. This selective propagation helps to maintain the richness of node representations even in deeper networks.

### 4.1.1 Key Contributions

This chapter presents the following key contributions in resource allocation in fog computing:

- Proposed GGCN: Introduces Gated Graph Convolution Neural Networks (GGCN) as a novel resource allocation mechanism to optimize application loop delay and total network usage, addressing network congestion and performance issues.
- Comparative Analysis: Conducts a comparative analysis with benchmark algorithms, including FCFS, SJF, and PSO. The results demonstrate the superiority of GGCN in terms of performance and resource allocation efficiency.
- Experimental Validation: Extensive validation using the iFogSim simulator demonstrates the effectiveness of the proposed GGCN model. It showcases a noticeable reduction in delay and bandwidth consumption.
- Resource Optimization: Highlights the ability of GGCN to optimize resource allocation by minimizing node utilization without compromising performance, further improving its overall efficiency.

### 4.1.2 Overview of GGCN

Scholars have investigated the effectiveness of GGCN in versatile domains, for instance:

- Building Footprint Extraction:
  In recent research, GGCN has been successfully applied to enhance the semantic segmentation of building footprints from aerial imagery. By integrating GCNs with deep structured embedding, GGCN has refined weak and coarse semantic predictions into accurate detail boundaries, significantly improving the segmentation accuracy. This successful application of GGCNs in a real-world scenario is a testament to the technology's capabilities and potential [142].
- Automatic Modulation Identification (AMI):
  In a wireless communications study, GGCNs were employed to improve classification accuracy in low Signal-to-Noise Ratio environments by leveraging the

characteristics of received signals' temporal and embedded signaling features. This approach succeeded because GGCNs' architecture processes sequences of symbols and captures dependencies over time, which improves the detection of modulation types [143].

- Program Verifications:
  In the program verification study, GGCN was used to infer program invariants that approximate the set of program states reachable during execution. By representing the memory states as graphs and employing GGCNs, researchers can predict the logic formula describing the program's data structures, thus improving memory safety and correctness of program logic [144]

GGCNs display several strengths due to their robust architecture, which combines convolutional operations on graphs with the dynamic capabilities of gated units. This combination allows GGCNs to excel in scenarios where temporal and spatial data are crucial. In addition, the inclusion of residual connections in GGCNs enhances their ability to deep representations without suffering from vanishing gradients, making them more effective for complex tasks. Overall, Gated graph Convolutional Networks represent a powerful mechanism for handling graph-structured data across various applications. Their ability to integrate local and global data through gating mechanisms makes them particularly suitable for complex tasks that require detailed understanding and data processing.

## 4.2 Resource Scheduling Strategies

This section explores various heuristics and optimisation-based benchmark algorithms to analyze the performance of the proposed resource allocation mechanism. The terms **tuple**, **task**, and **job** are used interchangeably. These benchmark algorithms, including SJF, FCFS, and PSO, are modelled in the iFogSim toolkit to provide a comparative performance analysis against the proposed GGCN.

### 4.2.1 Shortest Job First

Algorithm 1 is SJF, prioritising incoming sensor requests $R_i \in S1, S2, \ldots Sn$, each associated with an estimated processing time. This approach can lead to improved QoS and better system performance compared to FCFS because it considers minimising waiting and response times. The SJF mechanism evaluates sensor requests based on their processing requirements and stores them in a waiting list $L_w$ if immediate processing is not possible. When the fog node becomes available, SJF allocates resources to the request $R_{sjf} \in L_w$ with the shortest estimated processing time. This ensures optimal use of fog resources by prioritising the quickest jobs.

**Algorithm 1** Shortest Job First

**Input:** Queue of sensor Requests: $S1, S2, \ldots Sn$, each associated with an estimated processing time

**Output:** Optimizing fog node allocation for sensor data processing

```
Initialize an empty waiting list L_w
do
if new sensor data request R_i arrives then
    if L_w = φ and Fog node is available then
        Allocate fog node to R_i
        Process R_i (analyze data, compute response)
        send command to actuator based on R_i
    else
        Append R_i to L_W
        Sort L_w in ascending order based on estimated processing time
    end if
end if
if current fog node finishes processing R_f then
    R_sjf ← Select the request from L_w with shortest estimated processing time
    Allocate fog node to R_sjf
    Process R_sjf send command to actuator
    Remove R_sjf from L_w
end if
while L_w is not empty do
    wait for the next fog node to become available
    Repeat resource allocation for the next request in L_w
end while
```

### 4.2.2 First Come First Served

Algorithm 2 is FCFS, one of the simplest algorithms for scheduling resources. In the context of our model, it handles sensor requests $R_i \in S1, S2, \ldots Sn$ in the order of arrival (FIFO). This mechanism ensures fairness, treating all requests equally without prioritising any over others. FCFS allocates fog node resources for all incoming sensors sequentially. If immediate processing is not possible, the requests are stored in a waiting list $L_w$ and handled in order. While this method is simple to implement, it may not provide optimal system performance or QoS.

**Algorithm 2** First Come First Served

**Input:** Queue of Sensor Requests: $S1, S2, \ldots Sn$
**Output:** sensor Requests allocated to fog nodes and commands sent to actuators

Initialize an empty waiting list $L_w$
do
**if** new sensor data request$R_i$ arrives **then**
    **if** $L_w = \phi$ and fog node is available **then**
        Allocate fog node to $R_i$
        Process $R_i$ (analyze data, compute response)
        send command to actuator based on $R_i$
    **else**
        Append $R_i$ to $L_W$
    **end if**
**end if**
**if** current fog node finishes processing $R_f$ **then**
    $R_{fcfs} \leftarrow$ Select the first request from $L_w$
    Allocate fog node to $R_{fcfs}$
    Process $R_{fcfs}$ and send command to actuator
    remove $R_{fcfs}$ from $L_w$
**end if**
**while** $L_w$ is not empty **do**
    wait for the next fog node to become available
    Repeat resource allocation for the next request in $L_w$
**end while**

### 4.2.3 Particle Swarm Optimization

Algorithm 3 is a PSO-based resource allocation strategy, an advanced optimisation technique inspired by the social behaviour of birds. It is a meta-heuristic and operates as a population-based approach. PSO is widely used in both cloud and fog computing for task scheduling purposes. It seeks optimal solutions for resource allocation among candidate solutions (Particles). Particles update their position based on their personal best experience $PBest_{id}$ and the global best experience $gBest_{gid}$. In this system model, PSO dynamically allocates fog nodes to incoming sensors, optimising based on response time, load balancing, and energy consumption. It is particularly suitable for complex, multi-dimensional optimisation problems in Fog computing.

Several researchers have applied modified or hybrid PSO approaches to improve their task scheduling, and it has also been extended to fuzzy control systems and ANN training. Most metaheuristic algorithms are inspired by natural or biological behaviours [145, 146, 147].

**Algorithm 3** Particle Swarm Optimization

**Input:** List of Fog nodes:$Fn1, Fn2, \ldots Fn$ and Sensor Requests:$Sn1, Sn2, \ldots Sn$

**Output:** Optimized fog node allocation for sensor data processing and actuator control

1: Initialise list of fog nodes with their capacities and current loads
2: Create an initial population of fog node allocation (particles), each representing a potential allocation configuration
3: Initialize the velocity and position of each particle based on their current state
4: Define fitness function to evaluate each particle based on criteria such as load balancing, response time, and energy efficiency and QoS
5: **repeat**
6: **for** each particle in swarm **do**
7: Compute fitness for the current allocation configuration
8: Update personal best (PBest) if current fitness is better than the best fitness found by this particle
9: Update global best (GBest) if current fitness is better than the best fitness found by any particle
10: Calculate new velocity for the particle based on cognitive and social components (influence from PBest and GBest)
11: Update position of the particle according to new velocity (adjust fog node allocation accordingly)
12: **end for**
13: Check for convergence criteria or maximum iterations
14: **until** convergence criteria met or maximum iterations reached
15: Deploy the optimized fog node allocation configuration for handling sensor data and actuator commands

Our system model uses PSO where particles iteratively update their velocity using:

$$V_{id} = WV_{id} + c1\ r1(PBest_{id} - X_{id}) + c2\ r2(gBest_{gid} - X_{id}) \tag{4.1}$$

where:

- $PBest_{id}$ is the personal best position of the particle $i$.
- $gBest_{gid}$ is the global best position across all particles.
- $c1$, $c2$ are positive constants(cognitive and social coefficients).
- $r1$, $r2$ are random values uniformly distributed in [0,1].
- $W$ is the inertia weight.

- $V_{id}$ is velocity of particle $i$ in dimension $d$.

The position of each particle is then updated as:

$$\bar{X}_{id} = X_{id} + V_{id} \tag{4.2}$$

Where $\bar{X}_{id}$ represents the new position of particle $i$ in dimension $d$, and $X_{id}$ is the current position of the particle.

This iterative process ensures convergence towards an optimal fog node allocation for efficient sensor data processing and actuator control.

## 4.3 Proposed GGCN Methodology

This section provides the considered system architecture as illustrated in Figure 4.1 and the gated graph convolution neural network (GGCN) is utilised for resource scheduling in fog computing environments. The system architecture comprises of the three layers such as Core, Edge and Access networks.

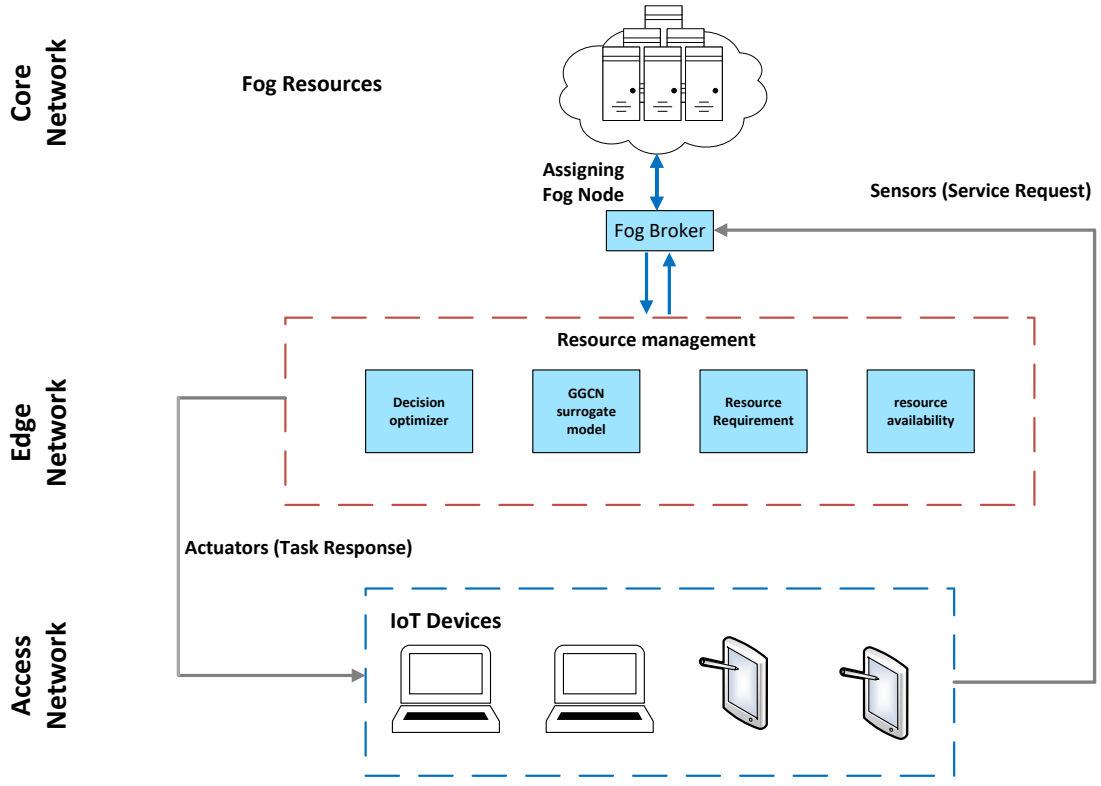


Figure 4.1: System Architecture

**The Core layer** represents a heterogeneous set of Fog resources, providing computational processes in close proximity to the IoT devices. This layer also encompasses cloud resources, serving as a centralised computing infrastructure that furnishes the system with storage and processing capabilities.

**The Edge layer** is the intermediate layer, located closer to the IoT devices and provides a low-latency communication channel. It comprises of a fog broker and a fog node.

- The fog broker is responsible for receiving sensor requests from the IoT devices. It acts as a mediator, initiating the resource allocation process by interacting with the resource management layer.
- Resource management includes a GGCN surrogate model, which predicts the system's performance based on the current resource allocation. The decision optimiser utilises this model, along with the binary indicator of node activity $x_n$, to optimise resource allocation, thereby enhancing QoS.
- The Fog nodes are represented by the set $F = \{F_1, F_2, \ldots, F_{|M|}\}$: where each fog node are assigned tasks by the fog broker. Each fog node $F_n$ (where $n \in M$) is active or not is governed by $x_n$ and contain the modules for task execution. Each $F_n$ manages a certain number of sensors $y_{n,s}$ constrained by capacity limits defined by $|M_{threshold}|$.

**The Access layer** comprises various IoT devices such as smartphones and tablets that generate tasks through sensors or other IoT devices which are represented by $\mathcal{S} = \{1, 2, \ldots, |S|\}$. $S$ are edge devices that collect data from environmental stimuli and generate tuples, while actuators provide a physical output based on computation in the fog. Actuators, similar to sensors, are edge devices and provide physical output in the environment based on the output of computation obtained in the fog node.

### 4.3.1 Problem Formulation

Let us assume that |S| and |M| are the total number of sensors and total number of fog nodes in the considered scenario, respectively, then

$$x_n = \begin{cases} 0, & \text{If node } F_n \text{ is not active} \\ 1, & \text{If node } F_n \text{ is active} \end{cases} \tag{4.3}$$

The variable $x_n$ is a binary indicator assuming the values of either 0 or 1 to show whether the fog node is active or not.

$$\sum_{n=1}^{M} x_n \leq |M_{threshold}| \tag{4.4}$$

The number of active fog nodes in the network are constrained such that no more than $|M_{threshold}|$ active fog nodes at any time. Let $y_{n,s}$ denote whether the sensor $s \in S$ is associated with the fog node $F_n$. $y_{n,s}$ is a binary indicator with a value of either 0 or 1 whereas $|S_n|$ is the maximum number of active sensors that can be managed by each fog node $F_n$.

$$\sum_{n=1}^{M} y_{n,s} \leq |S_n|, \forall s \tag{4.5}$$

Then, the total number of active sensors S in the complete network can be described as follows:

$$\sum_{n=1}^{M} x_n \times y_{n,s} \leq |\mathrm{S}|, \forall s \tag{4.6}$$

Let $D_{\text{avg}}$ demonstrate the average loop delay defined as the end-to-end latency across all modules in the system; it is expressed in milliseconds (ms) in iFogSim as in [148].

$$D_{avg} = \mathrm{CC} - \mathrm{ET}, \tag{4.7}$$

where CC symbolises the CloudSim Clock and ET denotes the emitting time of the tuple. Let $U_{\text{max}}$ denote the maximum bandwidth utilisation of the system, which is given by

$$U_{\text{max}} = \sum_{n=1}^{M} U_n, \tag{4.8}$$

where $U_n$ describes the network bandwidth utilisation of each fog node $F_n$. Our objective is to minimise $D_{\text{avg}}$ by optimising delays accrued in all modules that are assigned at each node $F_n$.

$$\begin{aligned} \min_{\mathbf{x_n}, \mathbf{y_{n,s}}} \quad & \tfrac{\omega}{D} \times D_{\text{avg}} + \tfrac{(1-\omega)}{U} \times U_{\text{max}} \\ \text{s.t.} \quad & 0 \leq \omega \leq 1 \\ & x_n \in \{0,1\}, \forall n, y_{n,s} \in \{0,1\}, \forall s \\ & (4.5) - (4.6) \end{aligned} \tag{4.9}$$

where $D$ and $U$ are the normalising coefficients representing the possible maximum system loop delay (in ms) and maximum system bandwidth utilisation (in bytes), respectively, calculated using an offline approach.

### 4.3.2 Proposed GGCN based Resource Scheduler

In this section, a new scheduling algorithm approach based on GGCN is proposed to solve the formulated problem in order to minimize the loop delay and the total network usage. The GGCN is a deep learning-based algorithm that leverages the power of neural networks to process data structured as graphs. It combines advanced graph embedding techniques with an anisotropic message-passing mechanism to make dynamic and efficient scheduling allocation decisions in fog computing environments. The proposed GGCN architecture (shown in Figure 4.2) consists of an initial input graph representation, followed by anisotropic message-passing through gating mechanisms (edge gates). The resulting embeddings are then updated using a Gated Recurrent Unit (GRU), which adaptively controls the updates, prevents vanishing gradients, and captures dynamic graph dependencies.

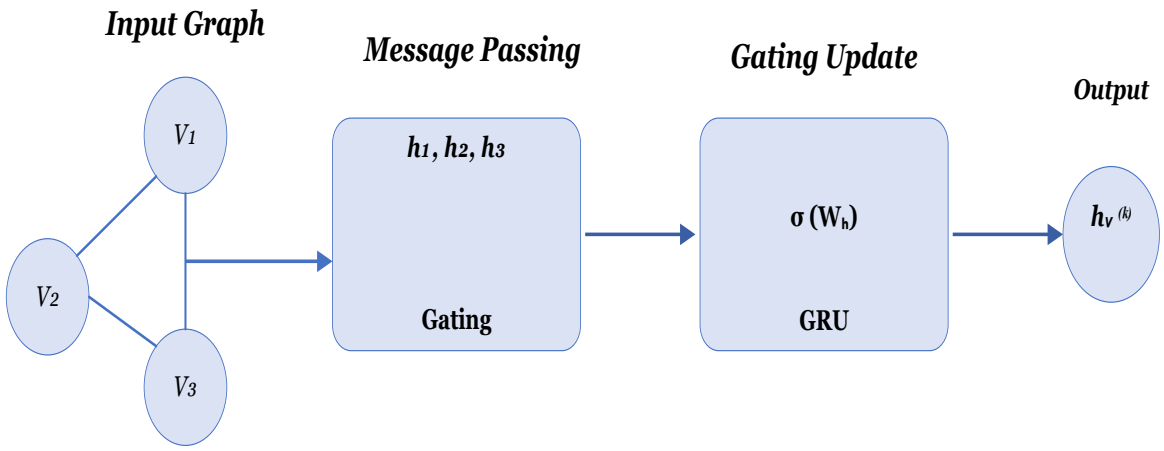


Figure 4.2: GGCN Architecture

In this architecture, the input graph represents a set of nodes (e.g., $v_1$, $v_2$, $v_3$), each corresponding to a task or resource. These nodes are connected based on their dependencies or similarities. The message passing block computes the intermediate embeddings ($h_1$, $h_2$, $h_3$) through a gating mechanism, which assigns importance weights to messages from neighbouring nodes. The gating update block then applies a Gating Recurrent Unit(GRU) using a learnable weight matrix ($W_h$) and a nonlinear activation function $\sigma$ to update each node's embedding. The output $h_v^{(k)}$ represents the updated embedding of node $v$ at iteration $k$.

In GGCN, the graphs can be represented as $G = (V, E)$, where $|V|$ and $|E|$ represent the set of vertices (nodes) and edges, respectively. A vertex corresponds

to a task or resource in a fog environment, while edges represent communication links or dependencies between nodes. The primary objective of GGCN is to classify and prioritise nodes by analysing their relationships and features. The edges are connections between nodes with a value of either zero or one. If nodes do not share similarities, the edge between them is zero; otherwise, the edge will be one.

Nodes with similar characteristics are linked, and their embeddings are updated through iterative neighbourhood aggregation, which enhances their representational power. During neighbourhood aggregation, the embedding for a node $v$ is updated by aggregating the embeddings of its neighbours $N(v)$, expressed as:

$$h_v^{(k)} = \sigma\left(W^{(k)} \cdot \text{AGGREGATE}\left(\{h_u^{k-1} : u \in \mathcal{N}(v)\}\right)\right),$$

where $h_v^{(k)}$ represents the embedding of node $v$ at iteration $k$, $W^{(k)}$ is the learnable weight matrix, and $\sigma$ is the activation function. This iterative mechanism enables the scheduler to capture both local and global structures, critical for efficient resource allocation. The GGCN architecture employs residual connections, batch normalisation, and edge gates to facilitate deeper network training while avoiding the vanishing gradient problem. These components ensure robust feature extraction and adaptability in processing large-scale graph data. Edge gates act as an attention mechanism, weighting the importance of dynamically connecting nodes.

To implement a GGCN as a resource allocation scheduler in iFogSim, the model aggregates sensors and resource information through graph embeddings and captures the spatial and temporal dependencies of sensors. These embeddings are then passed through multiple layers of the GGCN to generate sensor priorities and map the resources to active fog nodes. Finally, the output embedding guides the resource allocation decision, ensuring optimal allocation while adhering to the constraints in the problem formulation.

In the GGCN algorithm 4, the inputs $(S, V, E)$ are obtained from the resource manager. Graph $G = (V, E)$ represents sensors $(S)$ as vertices $(V)$ and their dependencies or resource connections as edges $(E)$. An estimated QoS score $\hat{O}$ is generated from the GGCN's learned embeddings, guiding the assignment of sensors to fog nodes. The scheduling decision splits the sensors and fog nodes into subsets of predetermined size (K). Sensors with higher QoS scores or tighter deadlines are prioritised. The embeddings of GGCN evaluate the resource availability at each fog node. If there are no fog nodes available, sensors are queued and the process iterates until all sensors are assigned to nodes.

**Algorithm 4** GGCN algorithm

**Input:** [S,V,E]
**Output:** Assigning sensors to fog node

**procedure** iFogSim
  $S, V, E \leftarrow$ ResourceManager()
  $G \leftarrow (V, E)$ ▷ generate graph representation of sensors and resources
  $\hat{O} \leftarrow$ GGCN (G) ▷ Learn embeddings and estimate QoS scores
  Sensor Subset $\leftarrow$ Extract Top $K$ sensors based on $\hat{O}$ and deadlines
  Fog Nodes $\leftarrow$ select nodes based on learned resource embeddings
  **for** $s \in$ Sensors subset **do**
    **if** allocation of $s$ to fog node is feasible based on embeddings **then**
      allocate $s \rightarrow$ Fog Node
    **else**
      Add $s$ to wait queue
    **end if**
  **end for**
**end procedure**

## 4.4 Performance Evaluation

This section presents the outcomes of the GGCN mechanism in various case studies, in comparison with the state-of-the-art mechanisms, including FCFS, SJF and a modified PSO. We carefully select two key performance metrics: the average loop delay and total network usage, which are critical indicators of QoS efficiency. Moreover, in this section, we focus on detailing our experimental setup, configuration parameters, different case studies, and results in detail. Figure 4.3 illustrates the general topology of our system, clearly summarising the number of clouds, gateways, fog nodes, sensors and actuators employed within the environment.

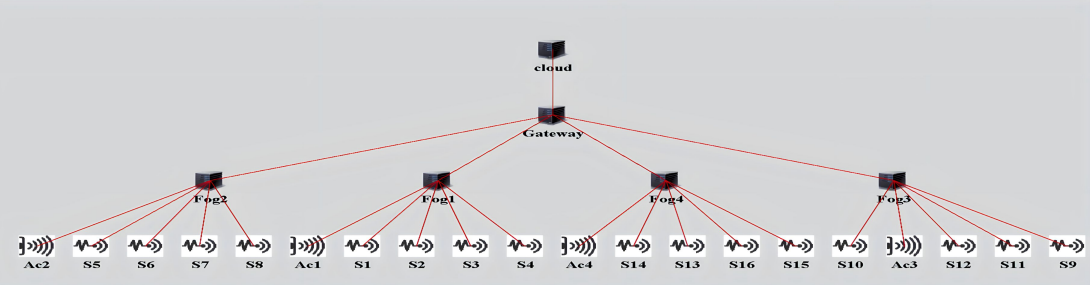


Figure 4.3: Fog Computing Topology Designed in iFogSim

### 4.4.1 Experiment Setup

For the valuation of our proposed deep learning scheduling approach, we developed our case study and implemented our approaches using extending some packages of iFogSim. Table 4.1 shows the simulation setup for our case study. Both our case study and algorithms were developed in Java using eclipse Ide and utilising iFogSim toolkit. iFogSim supports modelling and simulation for resource management techniques for IoT, edge and fog computing infrastructures.

Table 4.1: Simulation Setup

| system | Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz |
|---|---|
| Operating System | Windows 11 Home |
| Memory | 8 GB |
| Simulator | iFogSim |

### 4.4.2 Configuration

Table 4.2 illustrates the parameters of fog computing architecture configuration in our case study, including the cloud fog node and gateway. We consider three layers in our case study. The first layer is the cloud layer with specific specifications such as CPU length in million instructions per second, RAM in megabytes, uplink and downlink bandwidth in megabytes, level in fog computing architecture, cost per million instructions, busy power and idle power in watts. In our simulation, we keep the time interval transmission for each sensor as 5 ms, and the number of sensors per fog node is between 5 and 100. It is posited that the minimum and maximum fog node numbers are 1 and 4, respectively.

Table 4.2: Configurations

| Parameters | Cloud | Gateway | Fog node group 1 | Fog node group 2 |
|---|---|---|---|---|
| MIPS | 44800 | 2800 | 3000 | 2800 |
| RAM | 40000 | 4000 | 5000 | 4000 |
| UpBw | 100 | 10000 | 1200 | 1000 |
| DnBW | 10000 | 10000 | 11000 | 10000 |
| Level | 0 | 1 | 2 | 2 |
| Rate per MIPS | 0.01 | 0.0 | 0.0 | 0.0 |
| Busy Power | 16*103 | 107.339 | 107.339 | 107.339 |
| Idle Power | 16*103 | 83.4333 | 83.4333 | 83.4333 |

### 4.4.3 Case studies

Our hypothesis case study is a heterogeneous case study built on the iFogSim toolkit. We divided our fog node into two groups according to their location and specification. We did apply multiple approaches for the scheduling algorithm to map the module in suitable fog nodes and schedule incoming tasks in proper resources. The sensors are heterogeneous in their characteristics and are randomly linked to the fog node. To ensure consistency across different case studies, we established predetermined ranges of minimum and maximum sensors to be randomly linked to each node. Then, we conducted an inclusive system performance analysis by calculating the average number of sensors connected to each fog node for over 15,000 iterations. To ensure the performance of our proposed approach will be better than approaches if the load on the system has changed, we conducted six simulations for each case study, encompassing a range of sensor quantities, including 5, 10, 15, 30, 80 and 100.

In our case study, three application modules are utilized.

1. PROCESS DATA MODULE: operates at the fog node device, where it receives data from sensors and transmits system results to the gateway device.
2. FILTERING DATA MODULE: is designed to receive data from the process data module, filter and classify it, and enable training of the proposed GGCN.
3. DISPLAY DATA MODULE: The module stores or passes results to the fog node, enabling their transfer to the actuator.

Our case study set up a fog computing simulation in iFogSim, defining the hierarchical topology and application structure. It generally contains three main components:

- Physical Component: defines fog devices, sensors and actuators

- Logical Component: creates an application structure, defining modules and edges. Additionally, specifies tuple mappings and logical workflows for data flow between modules.
- Resource Management Component: Implements module mapping for processing in specific devices.

Some of the classes that would be used are introduced below:

- Sensors: simulates IoT sensors. The sensors generate tuples that transmit to the fog devices.
- Fog device: simulates and creates fog devices "fog nodes". Each fog node includes a specific memory, storage size, processor, uplink, and downlink. The fog nodes process the tuples based on scheduling decisions made by the fog broker.
- Tuples: Represents the packets of information passed across different locations in the iFogSim environment. Every tuple contains information about its source and destination.

The scenario extends certain classes from Module placements, specifically the placement policy "ModulePlacementEdgewards", to implement the proposed GGCN scheduler. This implementation serves as name Resource Manager, which dynamically allocates resources in active fog nodes using surrogate model and decision optimizer:

- GGCNLayer: Represents a layer in the Gated Graph Convolutional Networks (input, hidden, and output) layers which can:
  - Organize neurons and facilities embedding propagation.
  - Perform graph convolution for neighbor feature aggregation using adjacency matrix operations.
  - Ensure feature transformation through learnable weight matrices.
- Neuron: Defines individual neurons in the GGCN layers, which can handle:
  - Initializing neurons with weights and biases
  - Calculation gradients and updating weights during backpropagation.
  - interacting with edge gating mechanisms for dynamic connection weighting.
- GGCNUtil: Provides utility functions for GGCN operations, which can handle :

  - Sigmoid activation, its derivative and error calculation.
  - Random weight generation.
  - Edge gating functionality, including edge reweighting using EdgeGate.

- GGCNFogPlacment: Implements the GGCN-based resource scheduling allocation by
  - Generating Graph from fog device and sensor attributes to ensure adjacency matrix representation
  - Optimize module placement through training and inference.
  - Integrating anisotropic message-passing functionality from AnisotropicAggregator for directional feature propagation.

- Training: Represents training data for the GGCN, including:
  - Input attributes (latency, MIPS,..) and expected output.
  - Data preparation for embedding learning
  - Sampling techniques for batch-wise training

- SpatialTemporalEncoder: Encodes both spatial and temporal dependencies in the graph capable of:
  - Capturing dynamic relationships between sensors over time using time-decay functions.
  - Enhancing node embeddings with temporal awareness through attention mechanisms.

- OutputHead: generates final predictions from GGCN which by:
  - transforming embeddings into QoS scores and resource allocation decisions.
  - provides scheduling outputs, scoring mechanism “softmax”.

### 4.4.4 Results and Discussion

The results of our proposed approach focused on utilising the best mechanism to reduce the average loop delay with the total network usage of the system. In addition, our proposed algorithm is compared with benchmark algorithms such as FCFS, SJF and PSO approaches in terms of allocating proper resources to process incoming tasks. Figures 4.4, 4.5, 4.6, and4.7 illustrate the total network usage and average loop delay for different cases.

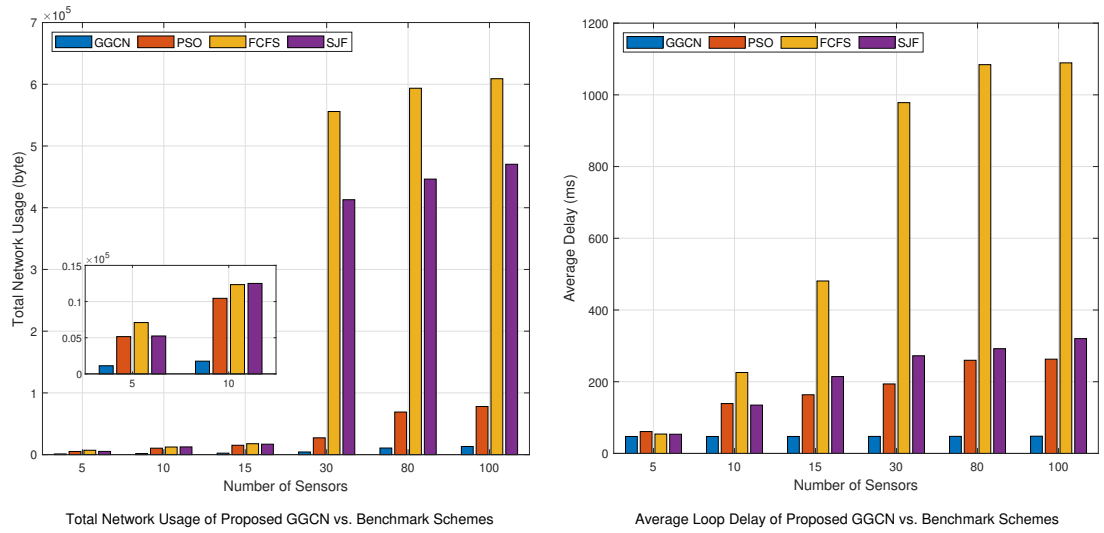


Figure 4.4: Performance Analysis of Proposed GGCN: Total Network Usage and Average Loop Delay Compared Benchmark Schemes (Case study A)

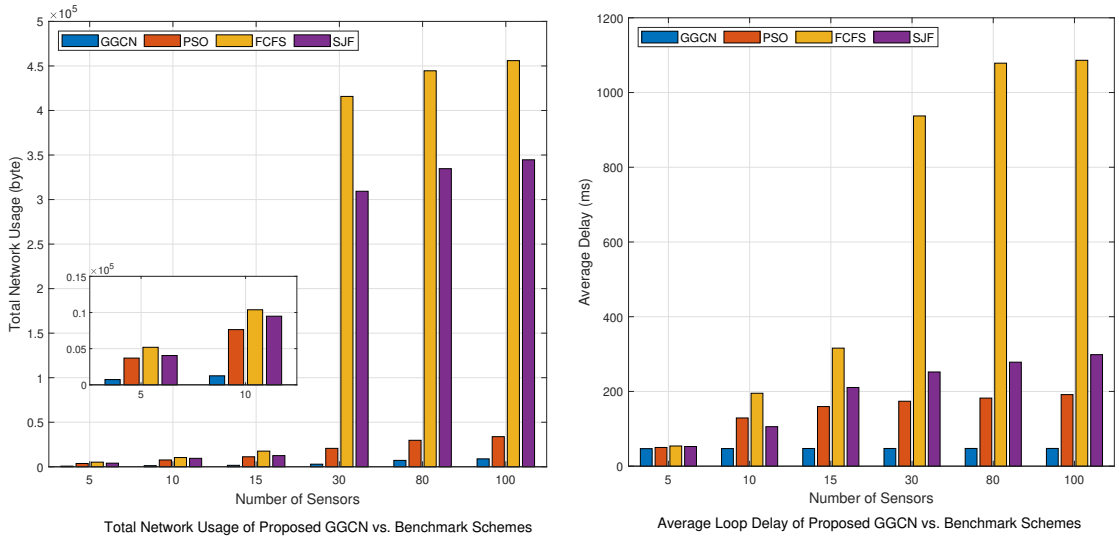


Figure 4.5: Performance Analysis of Proposed GGCN: Total Network Usage and Average Loop Delay Compared Benchmark Schemes (Case study B)

- Case study A: Four fog nodes with different capacities and dynamic random sensors are connected to those nodes using a random distribution method.
- Case study B: Three fog nodes with different capacities and dynamic random sensors are connected to those nodes using a random distribution method.

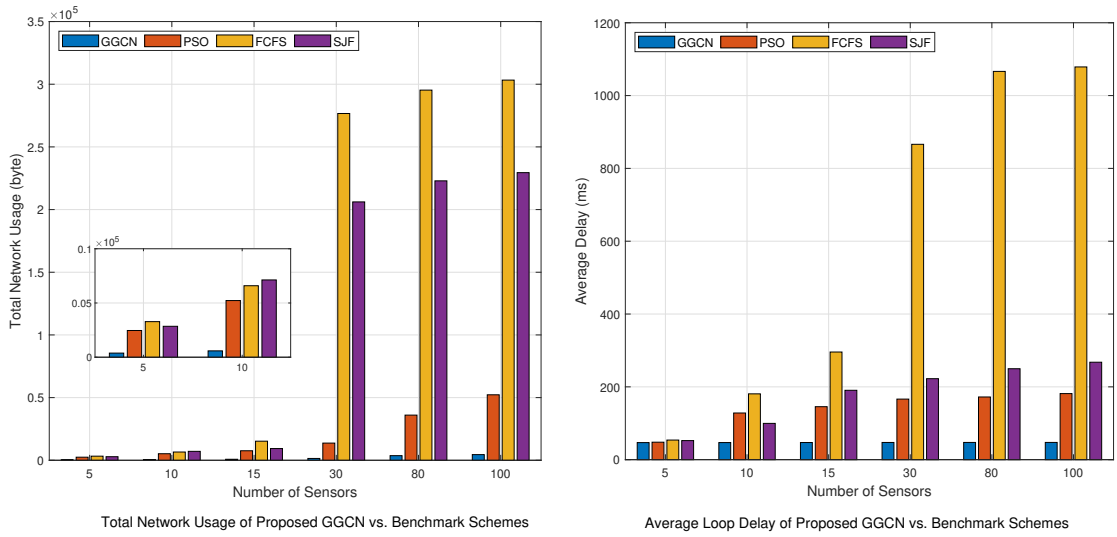


Figure 4.6: Performance Analysis of Proposed GGCN: Total Network Usage and Average Loop Delay Compared Benchmark Schemes (Case study C)

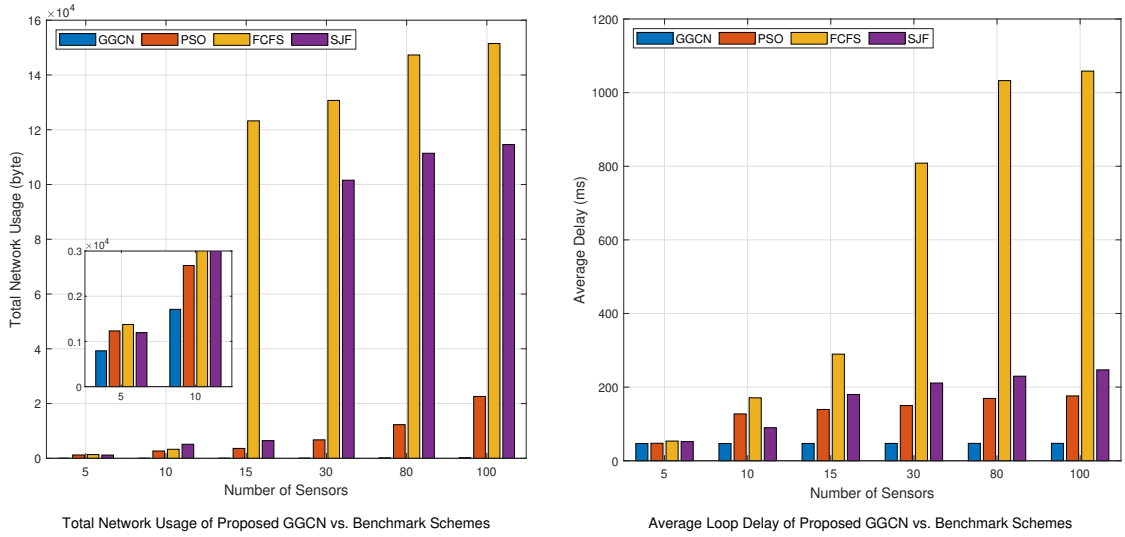


Figure 4.7: Performance Analysis of Proposed GGCN: Total Network Usage and Average Loop Delay Compared Benchmark Schemes (Case study D)

- Case study C: Two fog nodes with the same capacities and dynamic random sensors are connected to those nodes using a random distribution method.
- Case study D: One fog node dynamic with dynamic random sensors connected using a random distribution method.

Each of the above-mentioned cases is randomly linked to sensors of varying specifications, including 5, 10, 15, 30, 80 and 100 sensors. Our proposed algorithm consistently demonstrates a positive effect in reducing total network usage and directly impacting the average loop delay across all cases.

For total network usage, it is notable that four resource scheduling algorithms (GGCN, PSO, FCFS, and SJF) were evaluated in terms of their impact on network usage across different case studies (A, B, C, and D). It was observed that the total network usage generally increased as the number of sensors increased across all configurations regardless of the number of fog nodes used. However, the GGCN approach consistently recorded the lowest total network usage. In contrast, although SJF, PSO, and FCFS are efficient, their usage values are relatively higher. To illustrate, for case studies A with 100 sensors, SJF, PSO, and FCFS yielded 320.2 bytes, 262.9 bytes, and 1089.23 bytes, respectively, which are higher than the GGCN outcome. In the case of A, GGCN resulted in the lowest usage of 1122.9 bytes with only 5 sensors, and even with 100 sensors, the network usage remained significantly lower than the other approaches. In case studies D, the network usage ranged from 7.92 bytes to 190 bytes, depending on the number of sensors. The FCFS approach typically resulted in the highest usage, such as in the configuration with four fog nodes and 100 sensors.

On the other hand, evaluation of average loop delay, also termed as 'end-to-end latency', for all modules in the control loop across various configurations of fog nodes. This evaluation utilises four different resource scheduling algorithms - GGCN, PSO, FCFS, and SJF. Irrespective of the number of fog nodes and sensors, the GGCN approach consistently displayed superior performance by recording the lowest average loop delay. For instance, in case study A, GGCN achieved the minimum average loop delay of 47.25 ms with 5 sensors. As a point of comparison, both SJF and PSO strategies, while effective, did record higher average loop delays than GGCN. To illustrate, when operating with 100 sensors in case study A, SJF and PSO resulted in average loop delays of 320.2 ms and 262.9 ms, respectively, both exceeding the delay produced by the GGCN approach. FCFS too, under identical conditions, demonstrated the highest loop delay of 1089.23 ms. In case study B, GGCN recorded an average loop delay of 47.1 ms with 5 sensors which was significantly lower compared to 129.1 ms (PSO) and 195.16 ms (FCFS). Even with 100 sensors, GGCN maintained 47.48 ms, outperforming all benchmarks. For case study C, GGCN consistently delivered the lowest delay, recording 46.98 ms with 5 sensors compared to 128.2 ms (PSO) and 180.78 ms (FCFS). With 100 sensors, the delays increased to 47.39 ms (GGCN) and 267.67 ms (SJF). In case study D, GGCN continued to outperform, achieving an average loop delay of 46.72 ms with 5 sensors, while FCFS reported the highest delay of 1058.49 ms with 100 sensors. This trend, with GGCN delivering the lowest loop delay and FCFS the highest, persisted across all configurations - case

studies A, B, C, and D.

The ability of GGCNs to effectively capture the relationships between nodes via aggregated convolution filters is attributed to the performance achieved in optimising delay. This convolved aggregation enables the GGCN-based model to capture the dependencies between different data elements of nodes and permits it to learn a compressed node representation to preserve the relationships between the data elements while reducing the bandwidth utilisation during intra-node communication for task scheduling. This indicates that a GGCN-based proposed model learns a compressed representation of the data that captures the essential information while discarding redundant information, effectively reducing bandwidth utilisation.

## 4.5 Challenges and Limitations

Here are five special challenges encountered during the development and implementation stage:

1. Complexity of GGCN implementation: This represents the primary issue because fog nodes are resource-constrained, where all computational resources are limited, and the complexity of the neural network models, which requires frequent updates based on dynamic network conditions.

2. Scalability issues: When the number of sensors, IoT devices, and fog nodes increases, the computational process, memory requirements, and updating of the GGCN model also increase. This can lead to a violation of QoS and overall system performance. A GGCN-based scheduler must effectively manage available resources without violating QoS, especially delay and network usage.

3. Real-time Data Processing: Another challenge occurs because of the heterogeneous and dynamic nature of IoT and fog environments. The GGCN scheduler requires rapid adaptation to real-time conditions, including network load, resource availability, and nodes. Achieving this level of adaptation while maintaining low latency and high accuracy in scheduling decisions will make the GGCN mechanism a more efficient solution.

4. Integration with Existing System: Integrating neural network models with existing fog computing infrastructures creates several technical issues that require careful design and extensive simulation testing.

5. Energy Efficiency: Another critical aspect is implementing a GGCN scheduler to improve QoS without violating energy consumption. Reducing delay and

optimizing bandwidth are often linked to increased energy consumption. The proposed scheduler uses energy-aware resource allocation and adjusts node activity based on real-time demand. This approach reduces unnecessary energy usage while optimizing delay and bandwidth. Advanced prediction and learned capabilities prevent redundant computations and Idle power wastage is minimized. This balance ensures QoS improvements without compromising sustainability.

These are the main limitations of the proposed algorithm:

1. Performance in Simple Scenario: GGCN does not perform as well in simpler scenarios (e.g., single fog node) as it does in complex ones. The computational overhead may outweigh the benefits in such cases.

2. Energy Trade-offs: While GGCN optimizes delay and bandwidth, it demonstrates limited improvement in energy efficiency, particularly in low-resource utilization scenarios.

## 4.6 Conclusion

Increased adoption and utilisation of IoT devices present multiple performance challenges that require the use of new technology. This study proposed a GGCN mechanism to improve delay and total network usage in fog computing environments, showing promising results for integrating deep learning into fog computing environments. As the demand for efficient resource management in expanding IoT applications increases, the GGCN algorithm proposed in this study could become a cornerstone for future research and set a new standard for managing network performance. Further research should focus on enhancing scalability for larger networks, optimising the algorithm for specific IoT use cases, and exploring synergies with edge computing and blockchain for improved efficiency and security. The proposed GGCN approach significantly outperforms existing approaches. Specifically, it achieves:

- 86.09% improvement over PSO, 98.53% over FCFS, and 98.02% over SJF in total network usage.

- 68.64% improvement over PSO, 92.07% over FCFS, and 76.26% over SJF in the average loop delay across different case studies.

# Chapter 5

# eMLP-Based Task Scheduler to Optimize QoS in Fog Computing Environment

## 5.1 Motivation

The motivation behind this chapter is that task scheduling is a critical area in optimizing QoS due to limited and heterogeneous resources, and it encourages a further investigation into this area. Also, the majority of previous research focuses on applying evolutionary mechanisms or swarm intelligent algorithms such as PSO, GA, CS, ACA and several improved mechanisms. Additionally, the challenge of resource inefficiency in task scheduling is vital due to the complexity and heterogeneity of the tasks and fog nodes. The lack of investigation using deep neural network mechanisms motivates me to investigate this area and come up with a novel and dynamic approach to optimize QoS without violating system performance.

## 5.2 Overview

In this chapter, a proposed task-scheduling mechanism is introduced to tackle the current limitations in detail, designed to enhance the performance of applications in fog computing through a task-scheduling approach. As a key **contribution** of this research, the enhanced multilayer perception (eMLP) mechanism is introduce

Portions of this chapter are sourced from the paper listed in Publication Section as: *Enhancing QoS in Fog Computing via Enhanced Multilayer Perceptron-Based Multi-Objective Task Scheduling.*

to improve the performance of applications in fog computing via a task-scheduling approach. The following are the three main contributions of this chapter.

- Introduces a novel eMLP-based task-scheduling approach to QoS parameters.
- Minimizes average delay, bandwidth usage, power consumption, and resource costs while preventing over-provisioning and under-utilization.
- Validates eMLP performance through benchmarking against GNN, FCFS, and SJF, demonstrating its effectiveness.

The rest of this chapter is organized as follows: Resource Scheduling Strategies 5.3. MLP versatile Application in Section 5.3.5, and Problem Formulation in Section 5.4, The system Model in Section 5.5, and Performance evaluation in Section 5.6. Finally, The chapter concludes with a summary Section 5.7.

## 5.3 Resource Scheduling Strategies

Based on the literature review, several heuristics and optimisation-based benchmark algorithms will be used to compare and analyse the performance of our proposed task scheduling mechanism. The terms **tuple**, **task** and **job** have been used interchangeably. These benchmark algorithms, such as Shortest job First **(SJF)**, first come first served **(FCFS)**, Stable Matching **(SMA)**, and Graph Neural Network **(GNN)**, are utilized to provide a baseline to evaluate the performance and the effectiveness of our proposed algorithm in task scheduling within a fog computing environment.

### 5.3.1 Shortest Job First

As shown in Algorithm 5, the shortest job first mechanism executes the shortest tasks first. The first step on the SJF is to sort all incoming tuples $\mathbf{T}_i$ in the queue in ascending order. The first arrival tuple from the fog device executes immediately without restrictions, while subsequent tuples are executed based on their length. Tasks arrive at Fog devices from either Sensors or other fog devices. They are sent to the SJF Task scheduler. After that, the scheduler executes the first task, and other tasks are placed in the waiting queue $\mathbf{L}_{waiting}$. Every finished task $\mathbf{T}_m$ is moved to the finished tasks queue $\mathbf{L}_{finished}$. Then, SJF will select the shortest task from the waiting queue to be executed [47, 48].

**Algorithm 5** Shortest Job First

**Input** Task List(T1,T2,... $T_n$)
**Output** Executed Task List $L_{finished}$
Initialize $L_{waiting}$ to empty
**while** tasks are available or $L_{waiting} \neq \phi$ **do**
  **if** new task $T_i$arrives **then**
    Add $T_i$ to$L_{waiting}$
    Sort $L_{waiting}$ in ascending order of processing time
  **end if**
  **if** $L_{finished}$ is empty or current task completed **then**
    Select $T_m$ from $L_{waiting}$ with minimum processing time
    Assign $F_j$ (a facility or resource) to $T_m$
    Execute $T_m$
    Remove $T_m$ from $L_{waiting}$
    Add $T_m$ to $L_{finished}$
  **end if**
**end while**
**return** $L_{finished}$

### 5.3.2 First Come First Served

The first come first served mechanism 6 schedules the tasks based on the order arrival. It is simply like **FIFO** (First in First out) mechanism. The first task that arrives $T_i$ into fog devices is sent directly to the FCFS algorithm scheduler for execution. Subsequent tasks are stored in a waiting queue list $L_{waiting}$, in ascending order based on their arrival time. Tasks are then executed according to their arrival time from the waiting list. Like the SJF algorithm, the jobs (Tasks) arrive as tuples from sensors or other fog devices. Every finished task $T_m$ is added to the finished list $L_{finished}$[48]

### 5.3.3 Graph Neural Network

Graph Neural Network GNN is a deep learning neural network algorithm. It is designed to process data as graphs, making it suitable for complex systems with interrelated entities. The key features of GNN include graph-based input, node feature aggregation, iterative updates and flexibility in learning. GNN optimizes how tasks assignments to nodes by considering not only individual task requirements but also the dependencies between tasks. This makes GNN one of the sophisticated mechanisms for ensuring efficient and effective scheduling in distributed computing [149].

As shown in Algorithm 7, utilizes a graph-based approach to optimize task scheduling in fog computing. It processes a graph $G(V, E)$, where each node is V

represents a computational Task $T_i$, and each Edge E represents dependencies between these tasks. The GNN algorithm refines node representations of $T_i$ to drive optimal scheduling decisions by using iterative aggregation of features from neighboring nodes using propagating $\mathcal{P}$.

**Algorithm 6** First Come First Served

**Input** Task List(T1,T2,... $T_n$)
**Output** Executed Task List $L_{finished}$
Initialize $L_{waiting}$ to empty
Initialize $L_{finished}$ to empty
**while** not end of tasks OR $L_{waiting} \neq \phi$ **do**
  **if** new task $T_i$ arrives **then**
    **if** $L_{waiting} = \phi$ **then**
      Assign $F_j$ to $T_i$
      Execute $T_i$
      $T_m \leftarrow T_i$
      $L_{finished} \leftarrow T_m$
    **else**
      $L_{waiting} \leftarrow T_i$
    **end if**
  **end if**
  **if** $L_{waiting} \neq \phi$ and $L_{finished}$ is empty or current task completed **then**
    $T_{fifo} \leftarrow$ Select the first task from $L_{waiting}$
    Assign $F_j$ to $T_{fifo}$
    Execute $T_{fifo}$
    $T_m \leftarrow T_{fifo}$
    $L_{finished} \leftarrow T_m$
    Remove $T_{fifo}$ from $L_{waiting}$
  **end if**
**end while**
**return** $L_{finished}$

**Algorithm 7** Graph Neural Network GNN

**Input:** Graph G(V,E) with nodes representing tasks $T_i$ and edges $E$ representing dependencies.

**Output:** Optimized scheduling of $T_i$ on $F_j$.

Initialize each node $T_i \in V$ with task parameters
Construct G $V, E$ using tasks $T_i$ as nodes dependencies, $E$ as edge
Define propagation rules $\mathcal{P}$
**repeat**
    Aggregate features for each $T_i$ using $\mathcal{P}$
    Update $T_i$ features for scheduling
**until** Convergence or max_iter reached
Deploy the optimized task scheduling configuration based on $G$.

### 5.3.4 Stable Matching Algorithm

Stable matching algorithm 8 is a sophisticated scheduling mechanism designed to optimize task scheduling in fog computing environments. It systematically pairs tasks $T_i$ with fog nodes $F_j$ through an iterative proposal mechanism based on preference lists. Initially, all tasks will be placed in an unassigned tasks list. Each task proposes to its most preferred fog node, and nodes tentatively accept tasks based on their rankings, rejecting less preferred tasks as necessary.

**Algorithm 8** Stable Matching Algorithm SMA

**Input:** List of tasks $T_1, T_2, \ldots, T_n$

**Output:** Stable task-node pairs

Initialize all tasks $T_i$ and nodes$F_j$ as unassigned
**while** there exists an unassigned tasks $T_i$ **do**
    $T_i$ proposes to its most preferred node $F_j$ (not yet rejected)
    **if** $F_j$ is unassigned **then**
        $F_j$ tentatively accepts $T_i$
    **else if** $F_j$ prefers $T_i$ over its current task **then**
        $F_j$ rejects its current task and tentatively accepts $T_i$
    **else**
        $F_j$ rejects $T_i$
    **end if**
**end while**
**return** Stable task-node pairs

This mechanism continues until all tasks are stably matched or no feasible

assignments remain. SMA guarantees that each task is paired without violating system constraints, ensuring load balance and resource efficiency. [55].

### 5.3.5 MLP: Versatile Application

The versatility and effectiveness of MLP are unavoidable in the context of fog computing because it has demonstrated marvellous accomplishments in the field of medical surgery [150]. This is because the system contains very complex mechanisms that identify patterns and, further, MLP makes accurate predictions and thus is also used to solve many problems [151]. MLP integrated with a decision-tree algorithm is used in the diagnosis of many diseases, such as liver problems, etc. [152]. It is also used to manage energy consumption and to handle huge corpora and databases, these being its core characteristics [153]. Moreover, the technology is employed to schedule tasks in distributed computing technology [154]. Although it is very laborious and difficult to assign tasks to different nodes in cloud computing, because of the distributed, differentiated, and heterogeneous environment, MLP is capable of identifying patterns in order to assign tasks to fog nodes according to their capacities. It is hypothesized, therefore, that the application of MLP technology would ultimately maximize the merits of the algorithm so as to improve the quality of service. It is also flexible and adaptable, and thus has the potential to align with evolving technologies.

## 5.4 Problem Formulation

Assume a fog computing-based system has $N$ applications $A = \{A_1, A_2, A_3, \ldots, A_N\}$ running at the user's end, supported by $M$ heterogeneous fog nodes $F = \{F_1, F_2, F_3, \ldots, F_M\}$ for executing user applications. Each user application $A_y$ has $|k_y|$ independent tasks $T_{A_y} = \{T_{A_y,1}, T_{A_y,2}, T_{A_y,3}, \ldots, T_{A_y,|k_y|}\}$ that are offloaded to fog nodes. these tasks require network bandwidth, computing resources, and storage to execute on fog nodes with sufficient resources.
The Objective is to assign tasks from user applications to the most suitable fog nodes, minimizing average delay, reducing network usage and load, optimizing resource utilization (CPU, RAM, and storage), and minimizing power consumption.

The total number of tasks $N_T$ is:

$$N_T = \sum_{y \in A} |k_y| \tag{5.1}$$

The computational resources required are:

$$\sum_{y \in A} R_{\text{Req}}(T_{A_y}) \tag{5.2}$$

Where resources are quantified as CPU usage, RAM, Storage, and Bandwidth. Each task $T_i \in T_{A_y}$must be assigned to a fog node $F_j$. The available resources $R_{a,j}$ at a fog node $F_j$ are:

$$R_{a,j} = R_{tot,j} - R_{u,j} \tag{5.3}$$

Where $R_{tot,j}$ represents the total resources of fog node $F_j$, and $R_{u,j}$, represents the currently used resources. A task $T_i$ can only be assigned to fog node $F_j$ if:

$$R_{\text{Req}}(T_i) < R_{a,j} \tag{5.4}$$

Here $R_{\text{Req}}(T_i)$ represents the resource requirements( processor, memory, and disk) for task $T_{y,i}$ from fog node $F_j$. If multiple fog nodes satisfy this condition, optimization is applied to improve resource utilization, minimize delay, and optimize bandwidth and network load.

where $x_{i,j}$ is a binary decision variable defined as:

$$x_{ij} = \begin{cases} 1, & \text{If } T_i \text{ is assigned to } F_j, \\ 0, & \text{Otherwise.} \end{cases} \tag{5.5}$$

One of the objectives of this work is to manage task execution at fog nodes, managing both under-utilization and overloading. Fog nodes with significantly lower workloads than their capacity are penalized to minimize under-utilization, and those that exceed capacity are penalized to avoid overloading. The current utilization of each fog node is expressed as follows:

$$\textbf{(Obj 1:)} \quad \text{Minimize} \quad \sum_{j \in F} \left( \frac{Q_j}{C_j} \right) \approx \sum_{j \in F} \left( \frac{R_{u,j}}{R_{tot,j}} \right), \tag{5.6}$$

where $R_{u,j}$ is the resources already being used at fog node $j$, and $R_{tot,j}$ denotes the capacity of the fog node $j$ (e.g. processing capacity, memory). Similarly, $Q_j$ and $C_j$ represent the current workload and capacity of fog node $j$, respectively. Minimizing these ratios promotes a conservative workload distribution, ensuring balance. Conversely, maximizing the ratios encourages an aggressive utilization, potentially leading to overloading.

The second objective aims to ensure tasks are assigned to nearby fog nodes, thereby reducing latency and communication overhead. This is expressed as:

$$\textbf{(Obj 2:)} \quad \text{Minimize} \sum_{i \in T_{A_y}} \sum_{j \in F} (x_{i,j} \times D_{i,j}) \tag{5.7}$$

where $D_{ij}$ represents the distance between the application node hosting task $T_{y,i}$ and fog node $F_j$. This objective also minimizes the overall task delay, including

propagation, processing, and queuing delays. The amount of bandwidth $B_{i,j}$ required for task $T_i$ to communicate with fog node $F_j$

$$B_{i,j} = \frac{DS_i}{Th_{s,j}}, \tag{5.8}$$

where $DS_i$ represents data size and $Th_{s,j}$ is transfer rate. The total delay $L_{i,j}$ for executing $T_i$ on fog node $F_j$

$$L_{i,j} = D_{pn} + D_{pr} + D_q + D_{tx} \tag{5.9}$$

where $D_{pn}$ represents the propagation delay,$D_{pr}$ is the processing delay,$D_q$ denotes the queuing delay, and $D_{tx}$ represents the transmission delay. The resource usage cost $R_{i,j}$ based on the resources consumed by $F_j$ for the $T_i$,

$$R_{i,j} = \text{ResourceCostFunction}\left(\text{CPU}_j, \text{RAM}_j, \text{Storage}_j\right) \tag{5.10}$$

The power consumption $P_{i,j}$ cost per unit time for executing the task $T_i$ by the fog node $F_j$.

$$P_{i,j} = \frac{E_{i,j}}{t_{i,j}}, \tag{5.11}$$

where $E_{i,j}$ represents energy consumption and $t_{i,j}$ represents the execution time of task $T_i$. The Third objective is minimizing the average delay across all tasks, ensuring tasks are assigned to fog nodes that are able to reduce delay.

$$\textbf{(Obj 3:)} \quad \text{Minimize} \quad \bar{L} = \frac{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j} \cdot L_{i,j}}{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j}} \tag{5.12}$$

where $\bar{L}$ represents the average delay across all tasks, and $L_{i,j}$ represents the delay if task $T_{y,i}$ is assigned to fog node $F_j$. The fourth objective ensures an even distribution of workload among fog nodes, promoting balance and avoiding overloading or under-utilization:

$$\textbf{(Obj 4:)} \quad \text{Minimize} \left( \max_{j \in F} Q_j - \min_{j \in F} Q_j \right) \tag{5.13}$$

This objective minimizes the difference between maximum and minimum workload across all the available fog nodes, $F_j \in F$. The fifth objective is to give priority to the fog nodes with the current smaller workloads to balance the tasks across all the available fog nodes instead of overloading some of the fog nodes as below:

$$\textbf{(Obj 5:)} \quad \text{Minimize} \sum_{j \in F} Q_j^2 \tag{5.14}$$

The sixth objective is to minimize the overall average power consumption as below:

$$\textbf{(Obj 6:)} \quad \text{Minimize} \frac{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j} \cdot E_{i,j}}{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j} \cdot t_{i,j}} \tag{5.15}$$

As mentioned earlier, we have multiple objectives so we formulate our problem as a multi-objective problem with each objective assigned a different weightage (or priority) as follows:

$$\begin{aligned} \text{Minimize} \quad & w_1 \left( \sum_{j \in F} \left( \frac{Q_j}{C_j} \right) \right) + w_2 \left( \sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j} \cdot D_{i,j} \right) + w_3 \left( \max_{j \in F} Q_j - \min_{j \in F} Q_j \right) + w_4 \left( \sum_{j \in F} Q_j^2 \right) \\ & + w_5 \left( \frac{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j} \cdot L_{i,j}}{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j}} \right) + w_6 \left( \frac{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j} \cdot E_{i,j}}{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j} \cdot t_{i,j}} \right) \end{aligned} \tag{16}$$

s.t.

$$\sum_{F_j \in F} x_{i,j} = 1, \forall i \in T_{A_y} \tag{5.16a}$$

$$\sum_{i \in T_{A_y}} \sum_{j \in F} x_{i,j} \cdot D_{ij} \leq D_{threshold} \tag{5.16b}$$

$$Q_j \leq C_j \quad \forall j \in F \tag{5.16c}$$

$$\sum_{k=1}^{K} w_k = 1, \tag{5.16d}$$

Eq. (5.16a), to ensure each task $T_i$ must be assigned to exactly one fog node. Eq. (5.16b) ensures that the total delay for assigning all tasks of a single application $A_y$ to fog nodes does not exceed the predefined delay threshold $D_{threshold}$, promoting to assignment to closer nodes with lower delay. In Eq. (5.16c), $Q_j$ is the current workload of fog node $F_j$ and $C_j$ is the capacity of fog node $F_j$ (processing, memory). In Eq. (5.16d) $K$ represents the total number of objective functions, which is equal to 6.

To ensure comparability between the objectives, we normalize these objectives functions by taking the difference between the maximum and minimum possible values

calculated using the offline method as follows:

$$\begin{aligned}\text{Minimize} \quad & w_1 \left( \frac{\sum_{j \in F} (\frac{Q_j}{C_j})}{R_{\max}} \right) + w_2 \left( \frac{\sum_{i \in T_A y} \sum_{j \in F} x_{i,j} \cdot D_{i,j}}{D_{\max}} \right) \\ & + w_3 \left( \frac{\max_{j \in F} Q_j - \min_{j \in F} Q_j}{Q_{\max}} \right) + w_4 \left( \frac{\sum_{j \in F} Q_j^2}{Q_{\max}^2} \right) \\ & + w_5 \left( \frac{\frac{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{ij} \cdot L_{ij}}{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{ij}}}{L_{\max}} \right) + w_6 \left( \frac{\frac{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{ij} \cdot E_{ij}}{\sum_{i \in T_{A_y}} \sum_{j \in F} x_{ij}}}{E_{\max}} \right),\end{aligned} \tag{5.17}$$

In order to count the number of fog nodes being utilised to serve the tasks, the indicator function counts the distinct columns where atleast one task is assigned to each fog node :

$$F_{\text{used}} = \sum_{j \in F} \left( \text{Indicator} \left( \sum_{i \in T_{A_y}} x_{i,j} > 0 \right) \right) \tag{5.18}$$

$$\text{Indicator}(I) = \begin{cases} 1, & \text{if } I \text{ is true (i.e. the inner summation is greater than 0)} \\ 0, & \text{if } I \text{ is false (i.e. the inner summation is 0)} \end{cases}$$

This function sums the binary indicator for each column, representing the number of fog nodes in use.

## 5.5 System Model

This section introduces our system model, as depicted in Figure 5.1. The model comprises several key components, and these components include:

The system model is built upon **user devices** and sensors that serve as the primary sources of data generation and task initiation. It is also called end-user, which includes low-level users represented by sensors and actuators that are responsible for collecting data from the environment and transferring them to advanced user levels, such as smartphones, tablets, and computers that can run applications that need substantial computational resources. These applications $A$ consist of multiple independent tasks $T_{A_y}$, which require offloading to fog nodes for execution. When these devices request tasks, they are sent to the fog tier for better performance. All task-processing procedures will occur in the fog tier for more efficient handling. The **fog tier** receives application requests, which include a set of tasks from user devices. Tasks $T_{A_y}$ will be processed and manipulated in this tier. Each $T_{A_y}$ requires specific

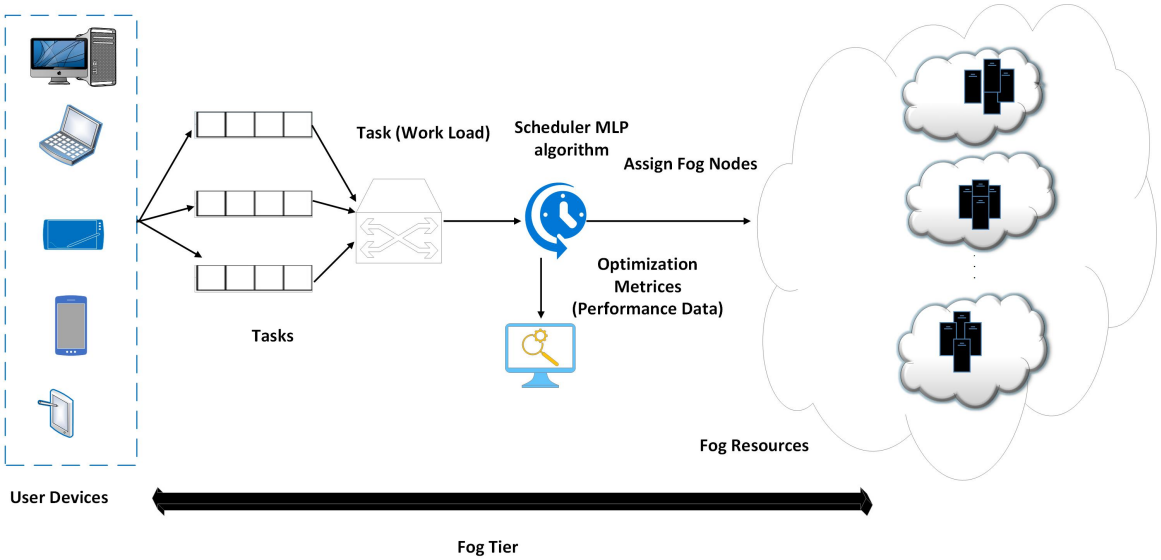


Figure 5.1: System Model

resources, such as processor cycles, memory, and storage to execute. These resources available $R_{a,j}$ are distributed among available fog node $F_j$, which are heterogeneous in their capabilities. All tasks and resource distribution will be served by the proposed eMLP scheduler to decide which are the suitable nodes to assign each task $T_{A_y}$. This scheduler acts as a decision-maker based on several factors, such as the complexity of tasks, memory requirements, and availability of resources in a fog node. The main aim of the proposed eMLP is to assign tasks to a suitable fog node $F_j$ without violating QoS parameters, while optimizing resource utilization $R_{a,j}$. The feedback loop in the proposed scheduler ensures that task assignment adapts dynamically by continually evaluating resource availability and system performance.

The outcome of the proposed model is to assign $T_{A_y}$ to $F_j$. All $F_j$ are geographically distributed in different locations and usually close to user devices, which gives them advantages in terms of minimizing delay and optimizing bandwidth usage. During task assignment, key parameters are considered, including the Distance $D_{i,j}$ (the physical distance between $T_{A_y}$ and $F_j$), bandwidth $B_{i,j}$ (the amount of network bandwidth consumed), and delay $L_{i,j}$ (the time required for task transmission and processing). These nodes are powerful in processing and handling tasks but have limited resources. A task will be allocated to an available fog node based on the proposed eMLP scheduler.

### 5.5.1 eMLP Scheduler

The proposed model is based on the eMLP scheduler, which assigns tasks to available and suitable nodes. To achieve this, the eMLP receives a vector of task attributes, including attributes such as memory requirements and the number of instructions. Each task is transformed within the eMLP's hidden layers, where weights $W$ and biases $b$ play pivotal roles, and Non-linearity is introduced in the model by utilising the ReLU activation function, which effectively enables the model to learn complex patterns. Weights are adjusted during the training phase to determine how task features affect the final output. A weight matrix $W^I$ in every layer $I^{th}$ will transform all input vectors from the previous layer $x^{(I-1)}$, which results in an intermediate result represented by $x^{(I)} = ReLU(W^{(1)} \cdot x^{(I-1)} + b^{(I)})$. The final step of getting the output of the final layer is acquired by applying the Softmax function to turn the results into probabilities in order to decide the allocation of tasks to the best-suited fog nodes.

Weights in the eMLP scheduler play a vital role in enhancing the scheduling tasks of the proposed scheduler. Initially, all weights in the neural network of the eMLP have a random set of values, and these will be adjusted during the back-propagation process. This process focuses on comparing predicted and actual task allocation, which will lead to adjusting the weights and enhancing the predictions of the eMLP. Adjustments apply the Adam optimization algorithm. It updates the learning rate for each weight based on estimates of the gradients for first and second moments. The updating weight is represented by $W_{new} = W_{old} - \frac{\alpha}{\sqrt{\hat{v}}+\epsilon}\hat{m}$ where $\hat{m}$ and $\hat{v}$ are biases to correct the first and second estimates of the gradients. $\epsilon$ is a small constant to avoid division by zero. This process will enable enhancing effectiveness during the training phase. Overall, the iteration of refinement processes will enable the eMLP scheduler to learn complex patterns by comparing task requirements and resource availability. It will increase the effectiveness of the fog network by optimising the distribution of computation loads in eMLP.

The algorithm 9 demonstrates the eMLP scheduler approach to efficiently assigning tasks to fog nodes. The algorithm begins by initializing the model with three hidden layers that are capable of handling complex tasks. Each task's feature vector is analyzed as input to the model. A comprehensive state $S_i$ represents the extracted feature and all preprocessing techniques. The eMLP computes the action probabilities $P_i$ through a deep network of eMLP. These probabilities indicate the optimal fog node for each task according to resource availability and the task size. The algorithm dynamically adjusts itself according to task changes, updating the learning phase to maintain accuracy. The eMLP scheduler continues to learn from real-time feedback.

This diagram 5.2 illustrates the implementation of eMLP for task-scheduling in a fog computing environment. Initially, there exist distinct tasks with unique attributes

**Algorithm 9** eMLP

**Input** Task features T1,T2,...$T_n$.
**Output** Task assignment A1,A2,...$A_n$.

1: Initialize eMLP with state_size, action_size,3 hidden layer
2: **for** each application $A_y$ **do**
3: **for** each task $T_i \in T_{A_y}$ **do**
**a.** Extract and preprocess features to form state $S_i$
**b.** Compute action probabilities $Pi$ using the deep network eMLP
**c.** Assign $T_i$ to $F_j$ with highest $Pi$
4: **end for**
5: **end for**
6: periodically updates eMLP with new data
7: monitor and adjust eMLP for performance

that must be allocated to a group of fog nodes identified as node-1 to node-m. The initial step involves the extraction of task features, which are then encoded in a suitable format for neural network processing. These features are used as the input for the eMLP. The eMLP's foundation is denoted by the label "hidden layer x 3", signifying the presence of ten hidden layers within the network. Each of the 24 neurons per hidden layer is represented by a yellow circle and utilizes the ReLU (Rectified Linear Unit) activation function to introduce non-linearity to the learning process. As the eMLP's network architecture is complex with ten hidden layers, the ReLU activation function is used instead of traditional ones, like Sigmoid, to boost training performance.

The output obtained from the hidden layers is then directed into an "output layer." Here, a "Softmax function" is applied to convert the output into a probability distribution. This helps in determining which fog node is most suitable for executing each task.

The diamond labelled "Decision" is the symbol for the decision mechanism. It uses the probabilities obtained from the Softmax function to choose the most appropriate fog node for each task, based on "Action Selection Based on Softmax Probabilities".

Furthermore, the diagram illustrates the training process by showing the flows of "backpropagation" and "feedforward", which are essential mechanisms in neural network learning. The "Categorical Crossentropy" loss function guides the backpropagation and evaluates the model's performance during training, allowing the weights to be updated for each training step to minimize losses.

Overall, the diagram provides a high-level overview of task scheduling using an eMLP within a fog computing architecture. It outlines the workflow from feature

extraction to the final decision-making process.

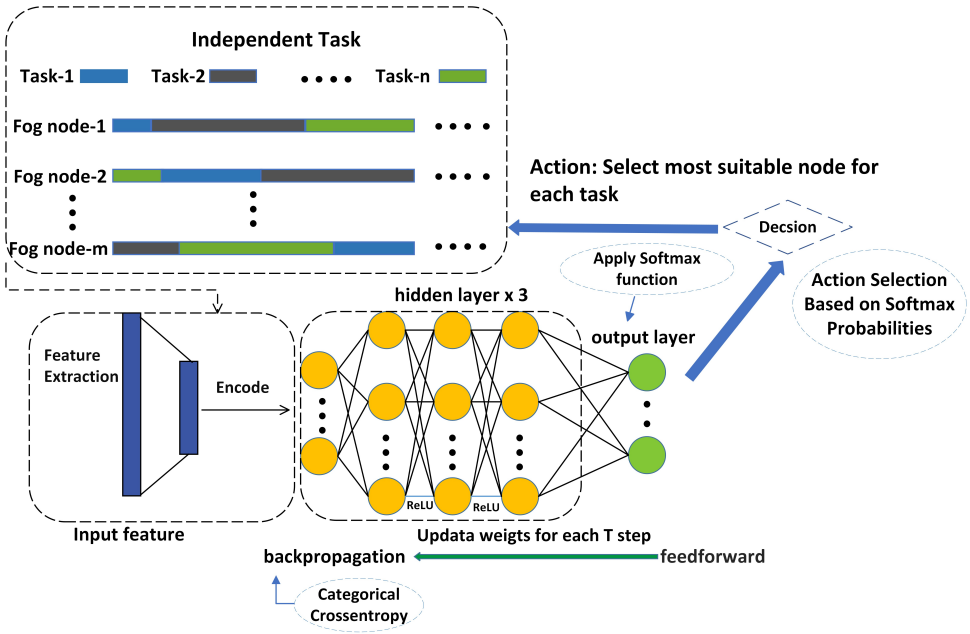


Figure 5.2: Systematic Block Diagram of the Proposed Enhanced Multilayer Perceptron (eMLP)

To summarise, the proposed eMLP scheduler extends the basic multilayer perceptron (MLP) by integrating advanced adaptive learning mechanisms to enhance task scheduling efficiency in a fog computing environment. The key difference between eMLP and a classic MLP can be summarised as follows:

- **Architecture:**
  - Classical MLP: Shallow, fewer hidden layers.
  - eMLP: Deeper architecture with multiple hidden layers(ten, 24 neurons per layer).
- **Activation Functions:**
  - Classical MLP: Typically uses simpler activation functions such as sigmoid.
  - eMLP: Utilises ReLU activation for superior performance and ability to capture complex patterns.
- **Learning and Adaptation:**

  - Classical MLP: Fixed behaviour post-training.
  - eMLP: Continuously learns and updates based on real-time feedback through ongoing backpropagation.

- **Decision-making:**
  - Classical MLP: Performs basic classification or regression.
  - eMLP: Implements Softmax-based probabilistic decision-making for optimised task assignment.

- **Optimization Techniques:**
  - Classical MLP: Employs basic gradient descent.
  - eMLP: Uses advanced optimisation through the Adam optimiser with adaptive learning rates, guided by the Categorical Crossentropy loss function.

These improvements enable the eMLP to dynamically respond to changes in task demands and resource availability to ensure accurate and context-aware scheduling decisions within fog computing environments.

## 5.6 Performance Evaluation

The Performance evaluation provides a comprehensive assessment of the proposed eMLP approach as a task scheduler in the fog computing environment. It examines eMLP's effectiveness compared to various other approaches. It also analyses the system configurations and resource distribution, highlighting the ability of eMLP's efficiency and adaptability to handle dynamic task sets.

### 5.6.1 Experimental Setup

This section describes our Experimental setup, which is built based on Fog computing architecture. Table 5.1 demonstrates the characteristics of our simulation, which includes system information, memory requirements, our operating system, and the simulator name.

Table 5.1: Simulation Setup

| | |
|---|---|
| system | Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz |
| Memory (RAM) | 8 GB |
| Operating System (OS) | Windows 11 |
| Simulator | Python (Leafsim) |

In our simulation, we did extend the Leafsim simulator to give us a more comprehensive analysis of our outcomes. We also consider the following predefined values for our simulation to prioritize resource utilization and delay minimization while maintaining a balanced focus on other objectives. The values assigned to the objectives in the Case One (CS1) are as follows: $w_1 = 0.3$, $w_2 = 0.2$, $w_3 = 0.25$, $w_4 = 0.15$, $w_5 = 0.05$, and $w_6 = 0.05$. These weights are designed to emphasize resource utilization and workload balancing as primary objectives with less importance to power consumption and delay to reflect their relative impact in this study.

In the Second Case Study (CS2), the weights are updated to $w_1 = 0.05$, $w_2 = 0.10$, $w_3 = 0.15$, $w_4 = 0.10$, $w_5 = 0.50$, and $w_6 = 0.10$. This case shifts the focus toward delay minimization as the dominant factor, while still considering network usage and a number of migrations. To ensure adaptability, this case follows an adaptive scheduling approach, balancing efficiency and delay reduction.

For delay parameters, based on several studies concerning ideal delay, the maximum delay $D_{\max}$ is set to 100 ms, $D_{\min}$ is 2 ms, and the $D_{threshold}$ is 100 ms to ensure strict enforcement of delay constraints [21, 155, 156]. Energy consumption values are bounded between $E_{\min} = 5$ W and $E_{\max} = 100$ W, consistent with the typical operational range of fog nodes[138, 157]. For fog node parameters, the initial workload of each node $Q_j$ is initialized to 0, representing no tasks assigned at the start. The capacity of each node $C_j$ is predefined based on its computational and storage limits, ensuring $Q_j \leq C_j$ throughout the simulation to avoid overloading.

### 5.6.2 Configuration

Task and node characteristics have been sourced from the literature review, and they are benchmarks for their proposed study [56].After deep investigation, this dataset used for the following reasons:

- The dataset is derived from a peer-reviewed study, ensuring it follows standard practices in fog and cloud computing research.
- The dataset addresses resource heterogeneity and realistic constraints.
- It is a well-defined dataset in terms of task attributes (e.g. number of instructions, memory, input/output file sizes) and node characteristics (e.g. CPU rate, cost), which provide a robust benchmark for evaluating the proposed scheduling mechanism and comparing it with others.
- There is a lack of datasets, especially for generic case studies, and most of the studies generate a simple dataset as a demonstration. Given the lack of datasets in fog computing, this dataset suits my generic case study as it includes a generic bag of tasks with a variety of tasks sizes and requirements.

Table 5.2 shows our configuration, which includes ten fog nodes with various attributes. These attributes are processing capacity, which is measured in MIPS (millions of instructions per second) , CPU, memory, and bandwidth. These attributes are associated with the cost of usage, which is measured by Grid Dollars (G$). It is a virtual currency unit commonly used in simulation studies. It does not represent real-world money but it serves as a standardized metric for modelling and evaluating the cost of resource usages, such as CPU, Memory and bandwidth [56, 158].

Table 5.2: Characteristics of Fog Nodes

| | |
|---|---|
| Nodes number | 10 nodes |
| CPU rate | [550,1500] |
| CPU usage cost | [0.1,0.4] |

Application requests are divided into separate tasks, each having a specific set of attributes, such as the number of instructions, memory requirements, input and output file sizes. To account for variations in request workloads, we have generated six data sets with 40 to 280 bags of tasks. The attributes of each task have been randomly assigned based on the specifications of the task. This randomness has resulted in a wide range of task profiles, from those that are computationally intense to those that demand greater memory or bandwidth. This ensures that we cover a diverse set of potential real-world scenarios. These are tasks characteristics that define their behaviour, which include:

1. Total count of instructions that the task will execute.
2. Tasks also require a certain amount of memory to function properly.
3. Tasks have input and output data files whose sizes must be considered.
4. Each of these properties has a range of values, with instructions being measured in billions (denoted by $10^9$ of instructions) and file sizes in megabytes.

### 5.6.3 Results and Discussion

In this section, we summarize the results of our simulation experiments and highlight the performance of the eMLP model in task-scheduling within a fog computing setup. We analyze important metrics, such as average delay, bandwidth, and cost, to assess the efficiency of eMLP compared to traditional algorithms, like SJF, FCFS, and SMA as well as the newer GNN approach.

The figure 5.3 shows that the eMLP-based model demonstrates a significant efficiency advantage over different scheduling algorithms when analyzing delay. As the

number of tasks increases, traditional algorithms such as SJF and FCFS experience a considerable increase in average delay, with FCFS consistently displaying the highest average delay figures. On the other hand, the SMA and eMLP models show superior performance, with the eMLP model outperforming all of them in terms of the least average delay across all task volumes. Specifically, when the number of tasks is high (280), eMLP maintains an average delay of 0.09 ms, which is in stark contrast to the 984.30 ms and 1568.30 ms observed for SJF and FCFS, respectively. Additionally, SMA approach demonstrates impressive performance by maintaining lower delay compared to FCFS, SJF and GNN, but higher than eMLP. At 280 tasks, SMA recorded 38.49 ms. Also, the GNN approach shows better performance than traditional algorithms, with an average delay of 784.71 ms at 280 tasks, although it does not outperform SMA or eMLP. This illustrates the ability of the eMLP model to effectively categorize tasks and select the fog node best suited for each one.

Additionally, the graph represents a case study (CS2) where delay is given high importance compared to other objectives. The eMLP model continues to show a significant improvement over other approaches. As task volume increases, FCFS and SJF still experience substantial growth in average delay. FCFS maintain the highest values among all approaches. However, eMLP achieves a remarkable delay of 0.06 ms, while SMA follows with 15.8 ms as the best performance among other models. GNN remains more effective than SJF and FCFS by 250.45 ms at 280 tasks. These results confirm the ability of eMLP's to optimize task assignments effectively.

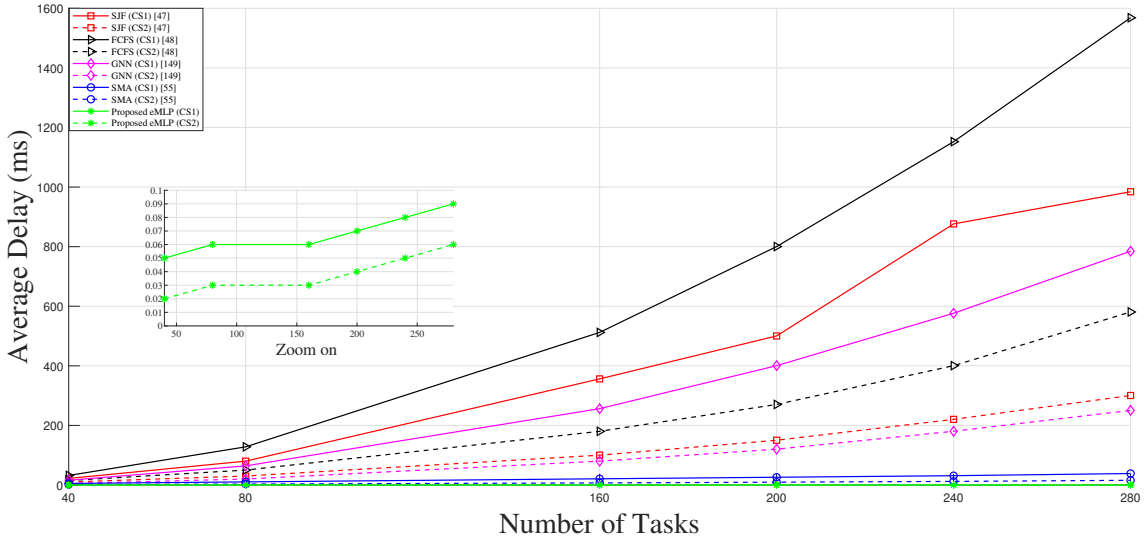


Figure 5.3: Performance Analysis of the Proposed eMLP Average Delay Compared to Multiple Benchmark Schemes Across Different Task Loads

Figure 5.4 shows network usage utilization, which is a measure of data transferred

in the fog network. The eMLP model shows the best result among all models. For example, in task 280, the Network usage is only 1.49 Mbps for eMLP in CS1, but 1568.50 for SJF, 1560.5 Mbps for FCFS, and 784.49 Mbps for GNN. Even though GNN demonstrates better network usage efficiency than FCFS and SJF, it is still less efficient than eMLP. Additionally, the SMA model shows competitive performance, recording 38.50 Mbps at 280 tasks, which is significantly lower than SJF, FCFS, and GNN but higher than eMLP.

For Case Study 2 (CS2), where network efficiency is given additional weight, eMLP further optimizes its usage, achieving 0.72 Mbps at 280 tasks, while GNN records 680.29 Mbps and SMA achieves 26.34 Mbps. Meanwhile, SJF and FCFS continue to perform poorly, with 1803.45 Mbps each. This highlights eMLP's ability to optimize network bandwidth while outperforming traditional and machine-learning-based models efficiently. The bandwidth parameter remains critical to optimize for several sectors where task throughput is essential.

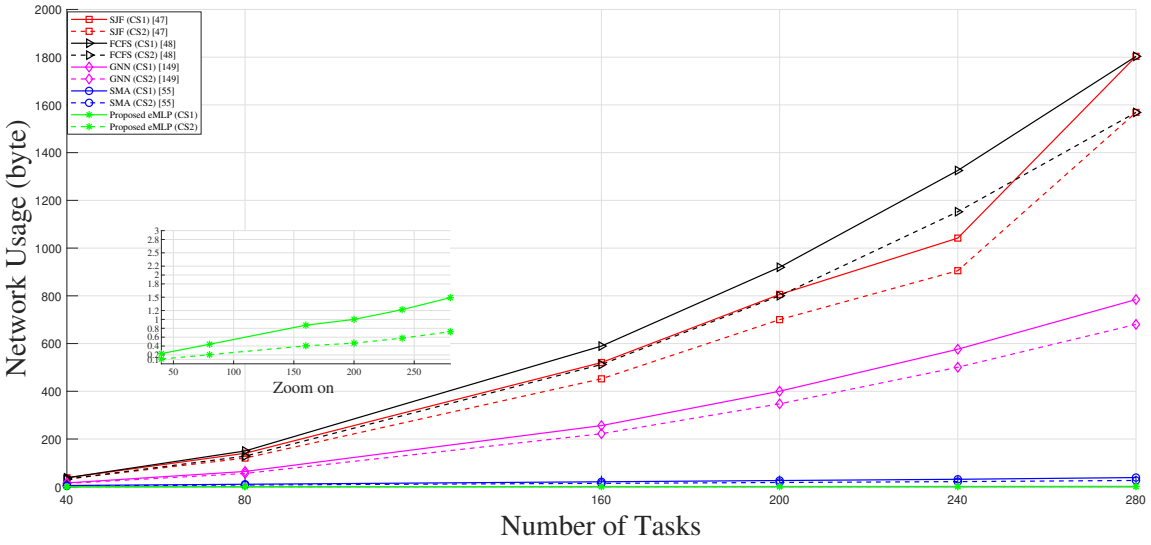


Figure 5.4: Performance Analysis of the Proposed eMLP Network Usage Compared to Multiple Benchmark Schemes Across Different Task Loads

The cost of resource usage is also critical because a fair balance must be struck between delay and network usage. Reducing it directly impacts other QoS parameters. The best solution is a trade-off between three parameters without violating them. Figure 5.5 shows how eMLP captures the cost, demonstrating that it has the lowest cost compared to other models. In task 280, the cost is 1.55 G$, 3922.29 G$, 4706.31 G$, 3138.40 G$ and 78.51 G$ for eMLP, SJF, FCFS, GNN, and SMA, respectively. In Case Study 2 (CS2), the eMLP model reduces cost to 0.583 G$ at 280 tasks, while SJF and FCFS remain high at 3482.34 G$ and 4201.35G$, respectively. GNN records

1942.56 G$, and SMA achieves 38.25 G$, showing a remarkable improvement but still higher than the eMLP model. These significant differences arise because eMLP effectively trades off between QoS parameters to achieve balance.

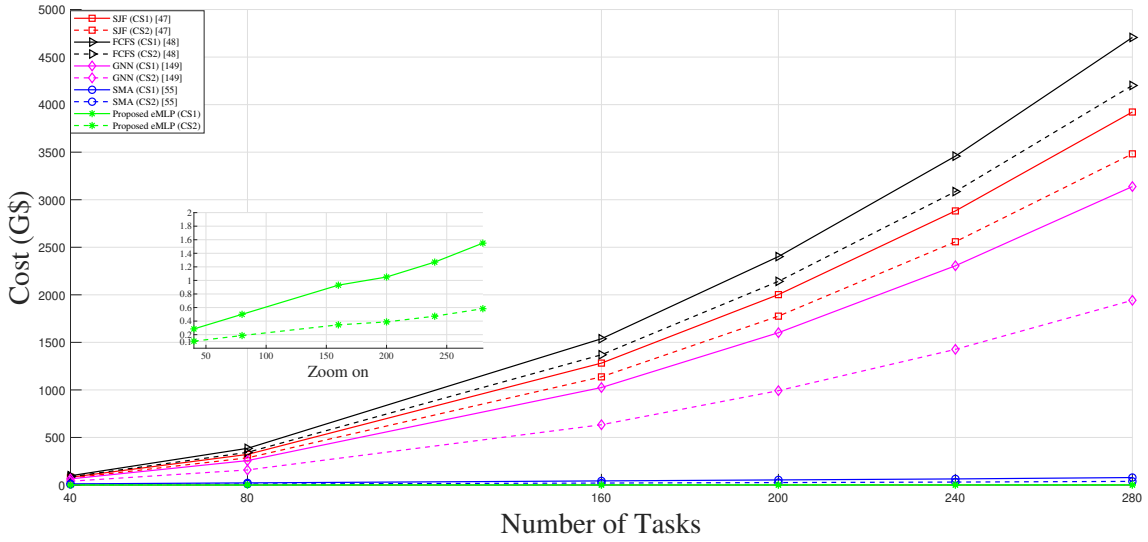


Figure 5.5: Performance Analysis of the Proposed eMLP Cost of Usage Compared to Multiple Benchmark Schemes Across Different Task Loads

Another parameter considered in this study is average power consumption, which directly impacts the cost and overall efficiency of the fog network. Figure 5.6 demonstrates the power consumption across different scheduling mechanisms. At 280 tasks in CS1, eMLP records the lowest power consumption among all approaches, with 5.0W. GNN is the second-best approach, consuming 36.10W less than SJF and FCFS, which consume 88.0W and 87.30W, respectively. Additionally, SMA reduces power consumption to 47.30W at 280 tasks, which shows better efficiency compared to traditional models.

For CS2, power consumption is further optimized compared to CS1. The eMLP model achieves a lower power consumption of 2.85W at 280 tasks, while GNN records 20.90W as the second-best model. The SMA model reaches 32.53W at 280 tasks. Meanwhile, SJF and FCFS maintain the highest power consumption reaching up to 74.58W and 74.51W, respectively. Overall, the results highlight the efficiency of eMLP in managing power consumption while maintaining optimal performance across other QoS parameters.

Figure 5.7 illustrates the percentage improvement of the proposed eMLP over benchmarks (SJF, FCFS, GNN, and SMA) across four key metrics: delay, bandwidth usage, cost of usage, and energy consumption. In all these metrics, the Proposed eMLP outperforms traditional scheduling schemes, showing significant improvement

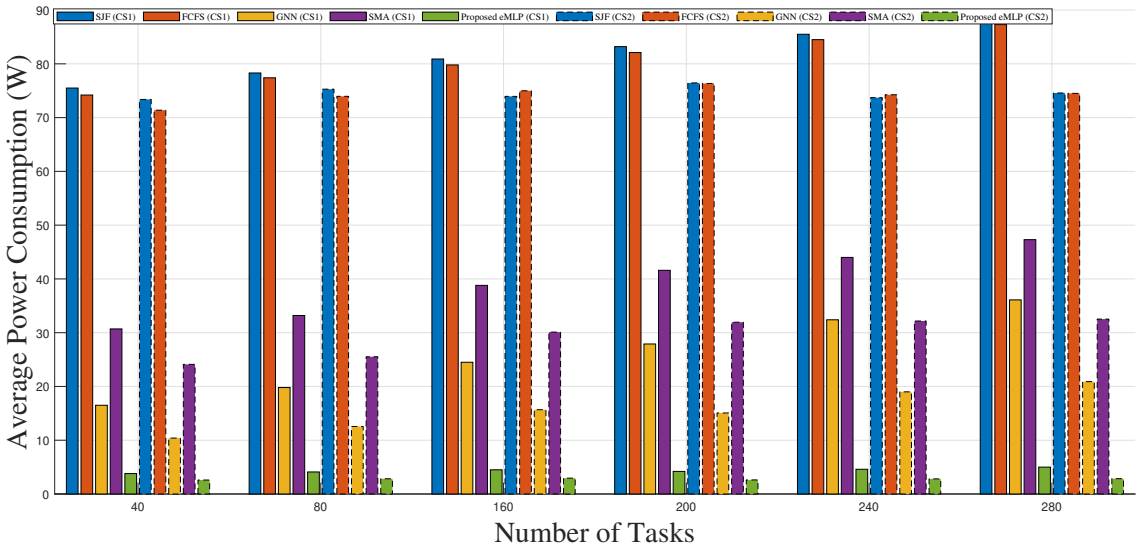


Figure 5.6: Performance Analysis of the Proposed eMLP Average Power Consumption Compared to Multiple Benchmark Schemes Across Different Task Loads

in resource efficiency during task scheduling.

For delay, eMLP achieves near-perfect optimization, with 99.8% improvement for all task loads, demonstrating its effectiveness in minimizing response time. Similarly, in bandwidth usage, the eMLP ensures minimal network overhead even under high task loads by reducing network resource consumption by over 95%. Regarding the cost of usage, eMLP shows both economic feasibility and scalability across different task loads, improving by over 99% compared to other mechanisms. In terms of energy consumption, eMLP outperformed other approaches by up to 85%, highlighting its suitability for energy-constrained environments like fog computing. These results underscore the eMLP's effectiveness in resource scheduling, its robustness, and adaptability for real-world applications.

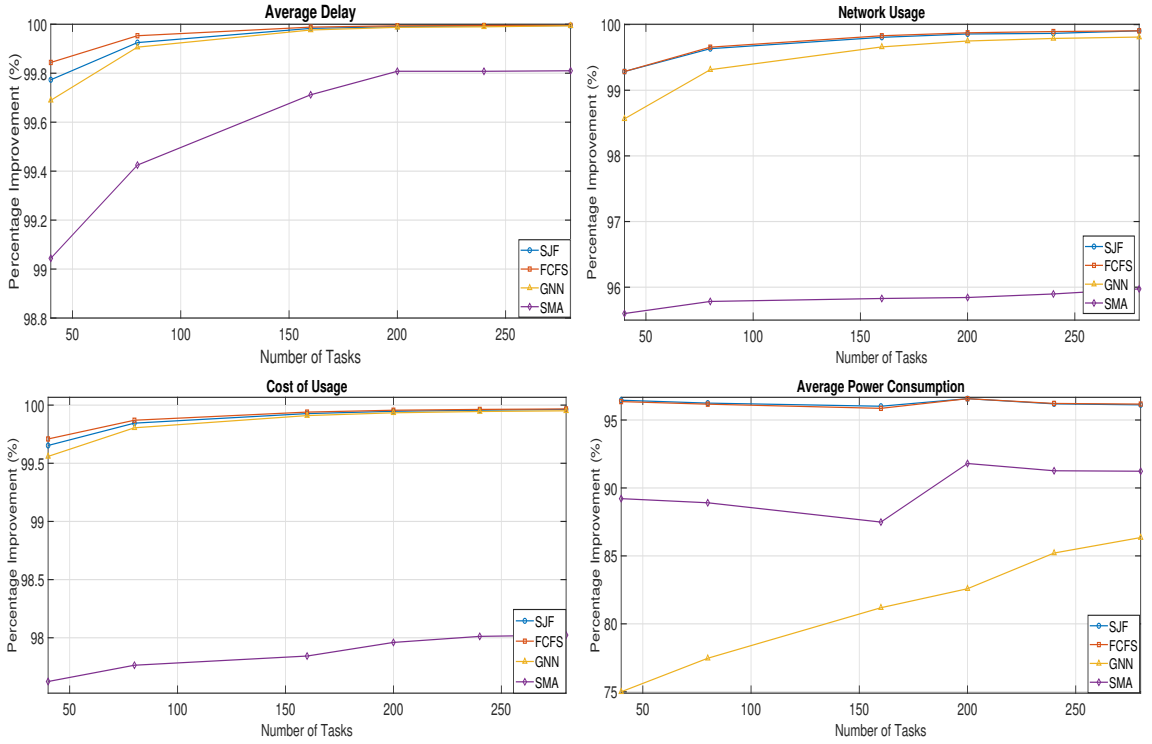


Figure 5.7: Percentage Improvement of the Proposed eMLP Over Benchmark schemes(FCFS, SJF, GNN, SMA)

## 5.7 Conclusion and Future Work

Due to the increase in IoT devices, several applications need a rapid response, and some of them need urgent responses to their requests. In this article, we have proposed eMLP, a novel task-scheduling approach in fog environments. The scheduler is able to minimize delay and cost of usage, while optimizing bandwidth usage. It was validated by comparing simulation results with state-of-the-art methods, including FCFS, SJF, SMA and GNN. In future work, the eMLP model will be applied in different domains as a task scheduler to test its effectiveness. It will include other performance metrics, such as energy consumption and deadlines.

# Chapter 6

# Conclusion and Future Directions

The thesis opens with an introduction the importance of QoS. It also outlines different distributed systems, including cloud, edge and fog computing. It demonstrates a comparison between these technologies, highlighting their objectives and limitations. In chapter 2, it details the resource management approach in the fog computing environment to optimize QoS. The thesis categorizes resource management issues into six categories including application placement, task scheduling, resource allocation, task offloading, load balancing, and resource provisioning. This chapter focus on task-oriented approaches which include application placement, task scheduling and Task offloading.

Chapter 3 discusses advanced resource management, which involves resource-oriented management including resource allocation, provisioning, and load balancing. Furthermore, this chapter provides insights into various simulation tools applicable in fog computing, providing comparative analyses of these tools. It also, addresses the limitations of resource nonengagement approaches.

Chapter 4 proposes a deep learning mechanism called gated graph convolution neural networks (GGCNs) for a novel resource scheduling management in fog computing, aimed at improving the average loop delay and optimizing the total network usage of the system. The GGCN approach significantly outperforms those of PSO (by 86.09%), FCFS (by 98.53%), and SJF (by 98.02%) in terms of total network usage. Additionally, GGCN surpasses PSO by 68.64%, FCFS by 92.07%, and SJF by 76.26% across all four considered scenarios in the average loop delay.

In chapter 5, proposes a new approach called enhanced Multi-Layer Perceptron (eMLP) for task scheduling in fog computing to reduce delays and cost and optimize bandwidth usage. The proposed mechanism was tested against several traditional algorithms, including First Come First Served, Shortest Job First, and Graph Neural Network approaches. It was tested in a bag of tasks applications in a fog computing environment. The experimental results show that enhanced MLP produced better

results: reducing latency by up to 75 %, optimizing bandwidth usage by approximately 65%, and increasing cost efficiency by 70%. The research concludes that eMLP is effective for task scheduling in fog computing, giving promising results to be investigate further in different sectors with different tasks constraints. Here is a list of future directions that require further investigation in the area of QoS:

- Security Enhancement in Fog Layers: Enhance security protocol within fog computing architectures to address vulnerabilities inherent in fog computing. Implement robust authentication mechanisms to ensure data integrity and privacy during access controls.

- Cross-Tier Optimization and Standardization: Implementing cross-tier optimization techniques that integrate fog computing with other network architectures like edge networks to enhance QoS. Develop standardized protocols and interfaces for seamless communication and resource sharing across these tiers. This approach boosts scalability and increases overall efficiency and reliability of services across the network.

- Enhanced QoS-aware Resource Management: Develop advanced, QoS-aware resource management frameworks that dynamically adapt to real-time conditions in services like virtual reality and online gaming, ensuring optimal performance across various IoT applications.

- Scalability in Resource allocation: Focus on scalable resource management strategies that can effectively accommodate the rapid growth of IoT devices and applications. Create adaptive algorithms that adjust up or down based on real-time demands and network conditions, maintaining optimal system performance.

- Energy-Efficiency Computing and Green Technologies: Concentrate on creating energy-efficient computing strategies that integrate green technologies. Implement mechanisms to reduce power consumption and utilize renewable energy sources for fog nodes. Design hardware and software solutions that prioritize energy efficiency. Strive to transform fog computing into a more sustainable technology that reduces its environmental impact while preserving or enhancing computational power.

- Energy-Efficient QoS Optimization: Develop advanced energy-aware resource allocation techniques that optimize delay and bandwidth while minimizing energy consumption. Investigate trade-offs between performance improvements and power efficiency to ensure sustainable QoS management in fog computing environments.

# References


[1] Kaneez Fizza et al. "QoE in IoT: a vision, survey and future directions". In: *Discover Internet of Things* 1 (2021), pp. 1–14.

[2] Sukhpal Singh Gill et al. "AI for next generation computing: Emerging trends and future directions". In: *Internet of Things* 19 (2022), p. 100514.

[3] Mariana-Daniela González-Zamar et al. "IoT technology applications-based smart cities: Research analysis". In: *Electronics* 9.8 (2020), p. 1246.

[4] Saber Talari et al. "A review of smart cities based on the internet of things concept". In: *Energies* 10.4 (2017), p. 421.

[5] Jagdeep Singh et al. "Fog computing: A taxonomy, systematic review, current trends and research challenges". In: *Journal of Parallel and Distributed Computing* 157 (2021), pp. 56–85.

[6] Sundas Iftikhar et al. "AI-based fog and edge computing: A systematic review, taxonomy and future directions". In: *Internet of Things* (2022), p. 100674.

[7] Moonmoon Chakraborty. "Fog Computing Vs. Cloud Computing". In: *Cloud Computing (May 3, 2019)* (2019).

[8] Michele De Donno, Koen Tange, and Nicola Dragoni. "Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog". In: *Ieee Access* 7 (2019), pp. 150936–150948.

[9] Mohit Kumar et al. "AI-Based Sustainable and Intelligent Offloading Framework for IIoT in Collaborative Cloud-Fog Environments". In: *IEEE Transactions on Consumer Electronics* (2023).

[10] Amal Al-Qamash et al. "Cloud, fog, and edge computing: A software engineering perspective". In: *2018 International Conference on Computer and Applications (ICCA)*. IEEE. 2018, pp. 276–284.

[11] Manoj Kumar Upadhyay, Mahfooz Alam, et al. "Edge Computing: Architecture, Application, Opportunities, and Challenges". In: *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*. IEEE. 2023, pp. 695–702.

[12] Blesson Varghese et al. "Challenges and opportunities in edge computing". In: *2016 IEEE international conference on smart cloud (SmartCloud)*. IEEE. 2016, pp. 20–26.

[13] Keyan Cao et al. "An overview on edge computing research". In: *IEEE access* 8 (2020), pp. 85714–85728.

[14] Sajeeda Parveen Shaik. "Strategic Placement of Servers in Mobile Cloud Computing: A Comprehensive Exploration of Edge Computing, Fog Computing, and Cloudlet Technologies". In: (2020).

[15] Pengfei Hu et al. "Survey on fog computing: architecture, key technologies, applications and open issues". In: *Journal of network and computer applications* 98 (2017), pp. 27–42.

[16] Hina Rafique et al. "A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing". In: *IEEE Access* 7 (2019), pp. 115760–115773.

[17] Vishal Kumar et al. "Comparison of fog computing & cloud computing". In: *Int. J. Math. Sci. Comput* 1 (2019), pp. 31–41.

[18] Flavio Bonomi et al. "Fog computing and its role in the internet of things". In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 2012, pp. 13–16.

[19] Yogeswaranathan Kalyani and Rem Collier. "A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture". In: *Sensors* 21.17 (2021), p. 5922.

[20] Wei Tian et al. "Telerobotic spinal surgery based on 5G network: the first 12 cases". In: *Neurospine* 17.1 (2020), p. 114.

[21] Akitoshi Nankaku et al. "Maximum acceptable communication delay for the realization of telesurgery". In: *PloS one* 17.10 (2022), e0274328.

[22] Jonathan Spruytte et al. "Planning omni-present networks of the future". In: *2014 Euro Med Telco Conference (EMTC)*. IEEE. 2014, pp. 1–6.

[23] Faris A Almalki et al. "Green IoT for eco-friendly and sustainable smart cities: future directions and opportunities". In: *Mobile Networks and Applications* 28.1 (2023), pp. 178–202.

[24] Md Masuduzzaman, Ramdhan Nugraha, and Soo Young Shin. "IoT-based CO 2 gas-level monitoring and automated decision-making system in smart factory using UAV-assisted MEC". In: *2022 International Conference on Decision Aid Sciences and Applications (DASA)*. IEEE. 2022, pp. 1023–1027.

[25] Jagruti Sahoo. "Cost-efficient, QoS and Security aware Placement of Smart Farming IoT Applications in Cloud-Fog Infrastructure". In: *arXiv preprint arXiv:2106.13524* (2021).

[26] Emad S Hassan et al. "Optimizing bandwidth utilization and traffic control in ISP networks for enhanced smart agriculture". In: *Plos one* 19.3 (2024), e0300650.

[27] Naif Alshammari et al. "Resource Scheduling in Integrated IoT and Fog Computing Environments: A Taxonomy, Survey and Future Directions". In: *Resource Management in Distributed Systems*. Springer, 2024, pp. 63–77.

[28] Mennan Selimi et al. "A lightweight service placement approach for community network micro-clouds". In: *Journal of Grid Computing* 17 (2019), pp. 169–189.

[29] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. "Latency-aware application module management for fog computing environments". In: *ACM Transactions on Internet Technology (TOIT)* 19.1 (2018), pp. 1–21.

[30] Mukhtar ME Mahmoud et al. "Towards energy-aware fog-enabled cloud of things for healthcare". In: *Computers & Electrical Engineering* 67 (2018), pp. 58–69.

[31] Mohit Taneja and Alan Davy. "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm". In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2017, pp. 1222–1228.

[32] Zoltán Adám Mann. "Decentralized application placement in fog computing". In: *IEEE Transactions on Parallel and Distributed Systems* 33.12 (2022), pp. 3262–3273.

[33] Leonan T Oliveira et al. "Enhancing modular application placement in a hierarchical fog computing: A latency and communication cost-sensitive approach". In: *Computer Communications* 216 (2024), pp. 95–111.

[34] Chun-Cheng Lin and Jhih-Wun Yang. "Cost-efficient deployment of fog computing systems at logistics centers in industry 4.0". In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4603–4611.

[35] Jaber Taghizadeh, Mostafa Ghobaei-Arani, and Ali Shahidinejad. "A metaheuristic-based data replica placement approach for data-intensive IoT applications in the fog computing environment". In: *Software: Practice and Experience* 52.2 (2022), pp. 482–505.

[36] BV Natesha and Ram Mohana Reddy Guddeti. "Meta-heuristic based hybrid service placement strategies for two-level fog computing architecture". In: *Journal of Network and Systems Management* 30.3 (2022), p. 47.

[37] Hemant Kumar Apat et al. "A hybrid meta-heuristic algorithm for multi-objective IoT service placement in fog computing environments". In: *Decision Analytics Journal* 10 (2024), p. 100379.

[38] H Sabireen and Neelanarayanan Venkataraman. "A hybrid and light weight metaheuristic approach with clustering for multi-objective resource scheduling and application placement in fog environment". In: *Expert Systems with Applications* 223 (2023), p. 119895.

[39] Karima Velasquez et al. "Service placement for latency reduction in the internet of things". In: *Annals of Telecommunications* 72 (2017), pp. 105–115.

[40] Malte Bellmann, Tobias Pfandzelter, and David Bermbach. "Predictive replica placement for mobile users in distributed fog data stores with client-side markov models". In: *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*. 2021, pp. 1–8.

[41] Gaurav Baranwal and Deo Prakash Vidyarthi. "FONS: a fog orchestrator node selection model to improve application placement in fog computing". In: *The Journal of Supercomputing* 77 (2021), pp. 10562–10589.

[42] Mohammad Aldossary. "Multi-layer fog-cloud architecture for optimizing the placement of IoT applications in smart cities". In: *Computers, Materials & Continua* 75.1 (2023), pp. 633–649.

[43] Mohammad Goudarzi, Marimuthu Palaniswami, and Rajkumar Buyya. "A distributed deep reinforcement learning technique for application placement in edge and fog computing environments". In: *IEEE Transactions on Mobile Computing* 22.5 (2021), pp. 2491–2505.

[44] Mansoureh Zare, Yasser Elmi Sola, and Hesam Hasanpour. "Towards distributed and autonomous IoT service placement in fog computing using asynchronous advantage actor-critic algorithm". In: *Journal of King Saud University-Computer and Information Sciences* 35.1 (2023), pp. 368–381.

[45] Supriya Singh and Deo Prakash Vidyarthi. "An integrated approach of ML-metaheuristics for secure service placement in fog-cloud ecosystem". In: *Internet of Things* 22 (2023), p. 100817.

[46] Yousef Abofathi, Babak Anari, and Mohammad Masdari. "A learning automata based approach for module placement in fog computing environment". In: *Expert Systems with Applications* 237 (2024), p. 121607.

[47] Bushra Jamil et al. "A job scheduling algorithm for delay and performance optimization in fog computing". In: *Concurrency and Computation: Practice and Experience* 32.7 (2020), e5581.

[48] Mxolisi Mtshali et al. "Multi-objective optimization approach for task scheduling in fog computing". In: *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*. IEEE. 2019, pp. 1–6.

[49] Xuan-Qui Pham and Eui-Nam Huh. "Towards task scheduling in a cloud-fog computing system". In: *2016 18th Asia-Pacific network operations and management symposium (APNOMS)*. IEEE. 2016, pp. 1–4.

[50] Valeria Cardellini et al. "On QoS-aware scheduling of data stream applications over fog computing infrastructures". In: *2015 IEEE Symposium on Computers and Communication (ISCC)*. IEEE. 2015, pp. 271–276.

[51] Sara Ghanavati, Jemal Abawajy, and Davood Izadi. "Automata-based dynamic fault tolerant task scheduling approach in fog computing". In: *IEEE Transactions on Emerging Topics in Computing* 10.1 (2020), pp. 488–499.

[52] Luiz F Bittencourt et al. "Mobility-aware application scheduling in fog computing". In: *IEEE Cloud Computing* 4.2 (2017), pp. 26–35.

[53] Yang Yang et al. "DEBTS: Delay energy balanced task scheduling in homogeneous fog networks". In: *IEEE Internet of Things Journal* 5.3 (2018), pp. 2094–2106.

[54] Jiafu Wan et al. "Fog computing for energy-aware load balancing and scheduling in smart factory". In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4548–4556.

[55] Ahmed S Alfakeeh and Muhammad Awais Javed. "Stable matching assisted resource allocation in fog computing based IoT networks". In: *Mathematics* 11.17 (2023), p. 3798.

[56] Binh Minh Nguyen et al. "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment". In: *Applied Sciences* 9.9 (2019), p. 1730.

[57] Aadharsh Roshan Nandhakumar et al. "EdgeAISim: A toolkit for simulation and modelling of AI models in edge computing environments". In: *Measurement: Sensors* 31 (2024), p. 100939.

[58] Shashank Swarup, Elhadi M Shakshuki, and Ansar Yasar. "Energy efficient task scheduling in fog environment using deep reinforcement learning approach". In: *Procedia Computer Science* 191 (2021), pp. 65–75.

[59] Fatma M Talaat. "Effective deep Q-networks (EDQN) strategy for resource allocation based on optimized reinforcement learning algorithm". In: *Multimedia Tools and Applications* 81.28 (2022), pp. 39945–39961.

[60] Nyoman Gunantara and I Nurweda Putra. "The characteristics of metaheuristic method in selection of path pairs on multicriteria ad hoc networks". In: *Journal of Computer Networks and Communications* 2019 (2019).

[61] Wan Norsyafizan W Muhamad et al. "Improvement of Energy Consumption in Fog Computing Via Task Offloading". In: *Journal of Advanced Research in Applied Sciences and Engineering Technology* 36.2 (2023), pp. 199–212.

[62] Yanwen Lan et al. "Task caching, offloading, and resource allocation in D2D-aided fog computing networks". In: *IEEE Access* 7 (2019), pp. 104876–104891.

[63] Wei Min et al. "Dynamic offloading in flying fog computing: optimizing IoT network performance with mobile drones". In: *Drones* 7.10 (2023), p. 622.

[64] Subhranshu Sekhar Tripathy, Sujit Bebortta, and Tanmay Mukherjee. "DynaFog: A Dynamic Task Offloading Framework for IoT-Based Fog Computing Platforms". In: *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE. 2024, pp. 464–469.

[65] Faizan Murtaza et al. "QoS-aware service provisioning in fog computing". In: *Journal of Network and Computer Applications* 165 (2020), p. 102674.

[66] Zhiwei Wei et al. "Many-to-many task offloading in vehicular fog computing: A multi-agent deep reinforcement learning approach". In: *IEEE Transactions on Mobile Computing* (2023).

[67] Amit Kishor and Chinmay Chakarbarty. "Task offloading in fog computing for using smart ant colony optimization". In: *Wireless personal communications* 127.2 (2022), pp. 1683–1704.

[68] Hoa Tran-Dang and Dong-Seong Kim. "Dynamic collaborative task offloading for delay minimization in the heterogeneous fog computing systems". In: *Journal of Communications and Networks* (2023).

[69] Xingxia Dai et al. "Task offloading for cloud-assisted fog computing with dynamic service caching in enterprise management systems". In: *IEEE Transactions on Industrial Informatics* 19.1 (2022), pp. 662–672.

[70] Sávio Melo et al. "OffFog: An approach to support the definition of offloading policies on fog computing". In: *Wireless Communications and Mobile Computing* 2022.1 (2022), p. 5331712.

[71] Yizhen Xu et al. "Task offloading for large-scale asynchronous mobile edge computing: An index policy approach". In: *IEEE Transactions on Signal Processing* 69 (2020), pp. 401–416.

[72] Liqing Liu et al. "Multiobjective optimization for computation offloading in fog computing". In: *IEEE Internet of Things Journal* 5.1 (2017), pp. 283–294.

[73] Moteb K Alasmari, Sami S Alwakeel, and Yousef A Alohali. "A Multi-Classifiers Based Algorithm for Energy Efficient Tasks Offloading in Fog Computing". In: *Sensors* 23.16 (2023), p. 7209.

[74] Jie Wang, Wenye Wang, and Cliff Wang. "Remedy or Resource Drain: Modeling and Analysis of Massive Task Offloading Processes in Fog". In: *IEEE Internet of Things Journal* 10.13 (2023), pp. 11669–11682.

[75] Francesco Chiti, Romano Fantacci, and Benedetta Picano. "A matching theory framework for tasks offloading in fog computing for IoT systems". In: *IEEE Internet of Things Journal* 5.6 (2018), pp. 5089–5096.

[76] Wan Norsyafizan W Muhamad et al. "Energy-efficient task offloading in fog computing for 5G cellular network". In: *Engineering Science and Technology, an International Journal* 50 (2024), p. 101628.

[77] Alireza Froozani Fard, Mohammadreza Mollahoseini Ardakani, and Kamal Mirzaie. "Multi-objective task offloading optimization in fog computing environment using INSCSA algorithm". In: *Cluster Computing* (2024), pp. 1–23.

[78] Guowei Zhang et al. "Fair task offloading among fog nodes in fog computing networks". In: *2018 IEEE international conference on communications (ICC)*. IEEE. 2018, pp. 1–6.

[79] Jean Lucas de Souza Toniolli and Brigitte Jaumard. "Resource allocation for multiple workflows in cloud-fog computing systems". In: *Proceedings of the 12th IEEE/ACM international conference on utility and cloud computing companion*. 2019, pp. 77–84.

[80] Xin Gao et al. "PORA: Predictive offloading and resource allocation in dynamic fog computing systems". In: *IEEE Internet of Things Journal* 7.1 (2019), pp. 72–87.

[81] Lina Ni et al. "Resource allocation strategy in fog computing based on priced timed petri nets". In: *ieee internet of things journal* 4.5 (2017), pp. 1216–1228.

[82] Raafat O Aburukba, Taha Landolsi, and Dalia Omer. "A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices". In: *Journal of Network and Computer Applications* 180 (2021), p. 102994.

[83] Onur Karatalay, Ioannis Psaromiligkos, and Benoit Champagne. "Energy-efficient resource allocation for D2D-assisted fog computing". In: *IEEE Transactions on Green Communications and Networking* 6.4 (2022), pp. 1990–2002.

[84] Keke Gai, Xiao Qin, and Liehuang Zhu. "An energy-aware high performance task allocation strategy in heterogeneous fog computing environments". In: *IEEE Transactions on Computers* 70.4 (2020), pp. 626–639.

[85] Jinghong Tan, Tsung-Hui Chang, and Tony QS Quelc. "Minimum energy resource allocation in FOG radio access network with fronthaul and latency constraints". In: *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE. 2018, pp. 1–5.

[86] Syed Rizwan Hassan et al. "Design of load-aware resource allocation for heterogeneous fog computing systems". In: *PeerJ Computer Science* 10 (2024), e1986.

[87] Yan Zhuang and Hui Zhou. "A hyper-heuristic resource allocation algorithm for fog computing". In: *Proceedings of the 2020 the 4th International Conference on Innovation in Artificial Intelligence*. 2020, pp. 194–199.

[88] Salim Bitam, Sherali Zeadally, and Abdelhamid Mellouk. "Fog computing job scheduling optimization based on bees swarm". In: *Enterprise Information Systems* 12.4 (2018), pp. 373–397.

[89] Bin Cao et al. "A resource allocation strategy in fog-cloud computing towards the Internet of Things in the 5g era". In: *2021 IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE. 2021, pp. 1–6.

[90] Saroja Subbaraj, Revathi Thiyagarajan, and Madavan Rengaraj. "A smart fog computing based real-time secure resource allocation and scheduling strategy using multi-objective crow search algorithm". In: *Journal of Ambient Intelligence and Humanized Computing* 14.2 (2023), pp. 1003–1015.

[91] Shahid Sultan Hajam and Shabir Ahmad Sofi. "Spider monkey optimization based resource allocation and scheduling in fog computing environment". In: *High-Confidence Computing* 3.3 (2023), p. 100149.

[92] Ismail Zahraddeen Yakubu and M Murali. "An Efficient IoT-Fog-Cloud Resource Allocation Framework Based on Two-Stage Approach". In: *IEEE Access* (2024).

[93] Samson Busuyi Akintoye and Antoine Bagula. "Improving quality-of-service in cloud/fog computing through efficient resource allocation". In: *Sensors* 19.6 (2019), p. 1267.

[94] Guoju Gao et al. "Auction-based VM allocation for deadline-sensitive tasks in distributed edge cloud". In: *IEEE Transactions on Services Computing* 14.6 (2019), pp. 1702–1716.

[95] Yutao Jiao et al. "Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks". In: *IEEE Transactions on Parallel and Distributed Systems* 30.9 (2019), pp. 1975–1989.

[96] Vibha Jain and Bijendra Kumar. "Auction based cost-efficient resource allocation by utilizing blockchain in fog computing". In: *Transactions on Emerging Telecommunications Technologies* 33.7 (2022), e4469.

[97] Nguyen Cong Luong et al. "A machine-learning-based auction for resource trading in fog computing". In: *IEEE Communications Magazine* 58.3 (2020), pp. 82–88.

[98] Vibha Jain and Bijendra Kumar. "Combinatorial auction based multi-task resource allocation in fog environment using blockchain and smart contracts". In: *Peer-to-Peer Networking and Applications* 14.5 (2021), pp. 3124–3142.

[99] Xiaosha Chen et al. "A machine-learning based time constrained resource allocation scheme for vehicular fog computing". In: *China Communications* 16.11 (2019), pp. 29–41.

[100] Bushra Jamil et al. "IRATS: A DRL-based intelligent priority and deadline-aware online resource allocation and task scheduling algorithm in a vehicular fog network". In: *Ad hoc networks* 141 (2023), p. 103090.

[101] Abdullah Lakhan et al. "Efficient deep-reinforcement learning aware resource allocation in SDN-enabled fog paradigm". In: *Automated Software Engineering* 29 (2022), pp. 1–25.

[102] Yihe Zhang et al. "Resource Allocation for Blockchain-Enabled Fog Computing with Deep Reinforcement Learning". In: *Proceedings of the 2022 12th International Conference on Communication and Network Security*. 2022, pp. 211–218.

[103] Fatma M Talaat. "Effective prediction and resource allocation method (EPRAM) in fog computing environment for smart healthcare system". In: *Multimedia Tools and Applications* 81.6 (2022), pp. 8235–8258.

[104] Jagdeep Singh et al. "An efficient machine learning-based resource allocation scheme for SDN-enabled fog computing environment". In: *IEEE Transactions on Vehicular Technology* 72.6 (2023), pp. 8004–8017.

[105] Simar Preet Singh et al. "Leveraging energy-efficient load balancing algorithms in fog computing". In: *Concurrency and Computation: Practice and Experience* 34.13 (2022), e5913.

[106] Ahmad Alzeyadi and Nazbanoo Farzaneh. "A novel energy-aware scheduling and load-balancing technique based on fog computing". In: *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE. 2019, pp. 104–109.

[107] Samah Ali and Raaid Alubady. "RWRR: Remind Weighted Rounding Robin for Load Balancing in Fog Computing". In: *2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*. IEEE. 2023, pp. 1–7.

[108] Jung-yeon Baek et al. "Managing fog networks using reinforcement learning based load balancing algorithm". In: *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2019, pp. 1–7.

[109] V Divya and R Leena Sri. "ReTra: reinforcement based traffic load balancer in fog based network". In: *2019 10th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE. 2019, pp. 1–6.

[110] Xiaolong Xu et al. "Dynamic resource allocation for load balancing in fog environment". In: *Wireless Communications and Mobile Computing* 2018.1 (2018), p. 6421607.

[111] Mohamed A Elsharkawey and Hosam E Refaat. "Mlrts: multi-level real-time scheduling algorithm for load balancing in fog computing environment". In: *International Journal of Modern Education and Computer Science* 11.2 (2018), p. 1.

[112] Jiafu Wan et al. "Fog computing for energy-aware load balancing and scheduling in smart factory". In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4548–4556.

[113] Wassim Boudieb et al. "Microservice instances selection and load balancing in fog computing using deep reinforcement learning approach". In: *Future Generation Computer Systems* 156 (2024), pp. 77–94.

[114] Muhammad Ibrahim, YunJung Lee, and Do-Hyuen Kim. "DALBFog: Deadline-aware and load-balanced task scheduling for the Internet of Things in fog computing". In: *IEEE Systems, Man, and Cybernetics Magazine* 10.1 (2024), pp. 62–71.

[115] Argyrios G Tasiopoulos et al. "Edge-MAP: Auction markets for edge resource provisioning". In: *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE. 2018, pp. 14–22.

[116] Nan Wang et al. "ENORM: A framework for edge node resource management". In: *IEEE transactions on services computing* 13.6 (2017), pp. 1086–1099.

[117] Adnan Khalid et al. "QoS based optimal resource allocation and workload balancing for fog enabled IoT". In: *Open Computer Science* 11.1 (2021), pp. 262–274.

[118] Francisco-Javier Ferrández-Pastor et al. "Deployment of IoT edge and fog computing technologies to develop smart building services". In: *Sustainability* 10.11 (2018), p. 3832.

[119] Masoumeh Etemadi, Mostafa Ghobaei-Arani, and Ali Shahidinejad. "A learning-based resource provisioning approach in the fog computing environment". In: *Journal of Experimental & Theoretical Artificial Intelligence* 33.6 (2021), pp. 1033–1056.

[120] Hojjat Baghban, Ching-Yao Huang, and Ching-Hsien Hsu. "Resource provisioning towards OPEX optimization in horizontal edge federation". In: *Computer Communications* 158 (2020), pp. 39–50.

[121] Nguyen Dinh Nguyen et al. "ElasticFog: Elastic resource provisioning in container-based fog computing". In: *IEEE Access* 8 (2020), pp. 183879–183890.

[122] Argyrios G Tasiopoulos et al. "FogSpot: Spot pricing for application provisioning in edge/fog computing". In: *IEEE Transactions on Services Computing* 14.6 (2019), pp. 1781–1795.

[123] Fan-Hsun Tseng et al. "A lightweight autoscaling mechanism for fog computing in industrial applications". In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4529–4537.

[124] Paola G Vinueza Naranjo, Enzo Baccarelli, and Michele Scarpiniti. "Design and energy-efficient resource management of virtualized networked Fog architectures for the real-time support of IoT applications". In: *The journal of Supercomputing* 74.6 (2018), pp. 2470–2507.

[125] José Santos et al. "Towards end-to-end resource provisioning in fog computing over low power wide area networks". In: *Journal of Network and Computer Applications* 175 (2021), p. 102915.

[126] Fabiana Rossi et al. "Geo-distributed efficient deployment of containers with Kubernetes". In: *Computer Communications* 159 (2020), pp. 161–174.

[127] Vadde Usha and TK Rao. "Resource provisioning optimization in fog computing: a hybrid meta-heuristic algorithm approach". In: *International Journal of System Assurance Engineering and Management* (2024), pp. 1–14.

[128] J.B. Awotunde, H.K. Tripathy, and A. Bandyopadhyay. "Hybrid Particle Swarm Optimization with Firefly based Resource Provisioning Technique for Data Fusion Fog-Cloud Computing Platforms". In: *FPA* 8 (2022), pp. 25–35. DOI: 10.54216/fpa.080203.

[129] Pejman Hosseinioun et al. "aTask scheduling approaches in fog computing: a survey". In: *Transactions on Emerging Telecommunications Technologies* (2020), e3792.

[130] Harshit Gupta et al. "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments". In: *Software: Practice and Experience* 47.9 (2017), pp. 1275–1296.

[131] Khaled Matrouk and Kholoud Alatoun. "Scheduling algorithms in fog computing: A survey". In: *International Journal of Networked and Distributed Computing* 9.1 (2021), pp. 59–74.

[132] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities". In: *2009 international conference on high performance computing & simulation*. IEEE. 2009, pp. 1–11.

[133] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. "Edgecloudsim: An environment for performance evaluation of edge computing systems". In: *Transactions on Emerging Telecommunications Technologies* 29.11 (2018), e3493.

[134] Fredrik Osterlind et al. "Cross-level sensor network simulation with cooja". In: *Proceedings. 2006 31st IEEE conference on local computer networks*. IEEE. 2006, pp. 641–648.

[135] Isaac Lera, Carlos Guerrero, and Carlos Juiz. "YAFS: A simulator for IoT scenarios in fog computing". In: *IEEE Access* 7 (2019), pp. 91745–91758.

[136] Tariq Qayyum et al. "FogNetSim++: A toolkit for modeling and simulation of distributed fog environment". In: *IEEE Access* 6 (2018), pp. 63570–63583.

[137] András Varga. "Discrete event simulation system". In: *Proc. of the European Simulation Multiconference (ESM'2001)*. Vol. 17. 2001.

[138] Philipp Wiesner and Lauritz Thamsen. "Leaf: Simulating large energy-aware fog computing environments". In: *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*. IEEE. 2021, pp. 29–36.

[139] Antonio Brogi and Stefano Forti. "QoS-aware deployment of IoT applications through the fog". In: *IEEE Internet of Things Journal* 4.5 (2017), pp. 1185–1192.

[140] Rodrigo N Calheiros et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". In: *Software: Practice and experience* 41.1 (2011), pp. 23–50.

[141] Naif Alshammari et al. "Delay and Total Network Usage Optimisation Using GGCN in Fog Computing". In: *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE. 2023, pp. 1–6.

[142] Yilei Shi, Qingyu Li, and Xiao Xiang Zhu. "Building segmentation through a gated graph convolutional neural network with deep structured feature embedding". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 159 (2020), pp. 184–197.

[143] Pejman Ghasemzadeh et al. "GGCNN: an efficiency-maximizing gated graph convolutional neural network architecture for automatic modulation identification". In: *IEEE Transactions on Wireless Communications* (2023).

[144] Yujia Li et al. "Gated graph sequence neural networks". In: *arXiv preprint arXiv:1511.05493* (2015).

[145] Mohamed K Hussein and Mohamed H Mousa. "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization". In: *IEEE Access* 8 (2020), pp. 37191–37201.

[146] Manoj Kumar Patel, Manas Ranjan Kabat, and Chita Ranjan Tripathy. "A hybrid ACO/PSO based algorithm for QoS multicast routing problem". In: *Ain Shams Engineering Journal* 5.1 (2014), pp. 113–120.

[147] Anita Singhrova and A Anu. "Prioritized GA-PSO algorithm for efficient resource allocation in fog computing". In: *Indian J. Comput. Sci. Eng.* 11.6 (2020), pp. 907–916.

[148] Amer Sallam et al. "Performance evaluation of fog-computing based on IoT healthcare application". In: *2021 International Conference of Technology, Science and Administration (ICTSA)*. IEEE. 2021, pp. 1–6.

[149] Liekang Zeng et al. "Serving Graph Neural Networks With Distributed Fog Servers for Smart IoT Services". In: *IEEE/ACM Transactions on Networking* (2023).

[150] Sami Alkadri et al. "Utilizing a multilayer perceptron artificial neural network to assess a virtual reality surgical procedure". In: *Computers in Biology and Medicine* 136 (2021), p. 104770.

[151] Marius-Constantin Popescu et al. "Multilayer perceptron and neural networks". In: *WSEAS Transactions on Circuits and Systems* 8.7 (2009), pp. 579–588.

[152] Moloud Abdar, Neil Yuwen Yen, and Jason Chi-Shun Hung. "Improving the diagnosis of liver disease using multilayer perceptron neural network and boosted decision trees". In: *Journal of Medical and Biological Engineering* 38 (2018), pp. 953–965.

[153] Sadegh Afzal et al. "Building energy consumption prediction using multilayer perceptron neural network-assisted models; comparison of different optimization algorithms". In: *Energy* 282 (2023), p. 128446.

[154] Arti Rana et al. "Application of multi layer (perceptron) artificial neural network in the diagnosis system: a systematic review". In: *2018 International conference on research in intelligent and computing in engineering (RICE)*. IEEE. 2018, pp. 1–6.

[155] Jonathan Deber et al. "How much faster is fast enough? user perception of latency & latency improvements in direct and indirect touch". In: *Proceedings of the 33rd annual acm conference on human factors in computing systems*. 2015, pp. 1827–1836.

[156] Shengmei Liu, Xiaokun Xu, and Mark Claypool. "A survey and taxonomy of latency compensation techniques for network computer games". In: *ACM Computing Surveys (CSUR)* 54.11s (2022), pp. 1–34.

[157] Sangeeta Kakati and Rupa Deka. "Computational and Adaptive Offloading in Edge/Fog based IoT environments". In: *2022 2nd International Conference on Intelligent Technologies (CONIT)*. IEEE. 2022, pp. 1–6.

[158] Rajkumar Buyya et al. "Economic models for management of resources in peer-to-peer and grid computing". In: *Commercial Applications for High-Performance Computing*. Vol. 4528. SPIE. 2001, pp. 13–25.