Delay and Total Network Usage Optimisation Using GGCN in Fog Computing

Naif Alshammari[1,2*], Haris Pervaiz[3], Hasan Ahmed[1], Qiang Ni[1]

[1]School of Computing and Communications, Lancaster University, Lancaster LA1 4YW, Uk
[2]Shaqra University, Shaqra Saudi Arabia
[3]University of Essex, Wivenhoe Park Colchester CO4 3SQ, UK
[*]Corresponding author: n.h.alshammari@lancaster.ac.uk

***Abstract*—Network performance and throughput is affected by network congestion, which is caused by unnecessary bandwidth over-utilisation, expanding transmission delays, and increase in cost. Fog computing has emerged as a promising solution to overcome these shortcomings by provisioning computational resources to the network's edge. However, selecting suitable fog nodes can pose challenges due to increased latency and high energy consumption, leading to unnecessary bandwidth utilisation. This study proposes a deep learning mechanism called gated graph convolution neural networks (GGCNs) for resource scheduling management in fog computing to improve the average loop delay and the total network usage of the system. Our deep learning mechanism promotes energy-efficient collaborative intelligence among IoT devices while optimising resource utilisation. Reducing energy consumption not only promotes but also enhances sustainability and scalability in IoT networks. Our proposed mechanism shows improved results compared with several benchmark algorithms, such as first come first serve, shortest job first, and particle swarm optimisation. Our results demonstrate that the proposed model will resolve the problem of application placement and present a noticeable reduction in delay and bandwidth. The results can prove to be a standard benchmark in the IoT-Fog computing discipline and used to enhance the quality of service in wide-ranging heterogeneous applications located at distributed locations.**

***Index Terms*—Fog computing, Quality of Service, Internet of Things, Gated graph convolution neural networks.**

I. Introduction

In the last few decades, the Internet of Things (IoT) and mobile devices have helped establish a digital identity for end users that enables them to collaborate, integrate, and manage computing resources. These devices are resource-deficient and their applications need to cope with the massive amount of data generated. Several cloud-based solutions are available to overcome these limitations.

However, these solutions are responsible for issues such as latency, response time, security, and privacy, which weaken the automated operations of the digital industry and processes. Middleware has emerged as an alternative solution to this problem as they reside between client devices and remote cloud and bring the cloud resources to the network's edge [1]. As a powerful middleware, fog computing has attracted the attention of researchers owing to its distributed architecture, efficient application module placement, improved efficiency, and reduced amount of data required to transfer to the cloud to store, process and analyse. Within the realm of fog computing, energy-efficient collaborative intelligence has emerged as a powerful strategy to improve resource optimisation and sustainability. Fog computing extends cloud computing resources by offloading the computation, data and application to the the network's edge near end-user [2]. This technology appears to overcome several issues in cloud computing such as delay, security and privacy. Fog computing processes data on edge devices, which will help reduce delay because only a low volume of data will be processed in the cloud. In comparison, cloud computing will process data in the cloud, which will lead to a significant delay because of the requesting process. Fog computing is considered a distributed framework that uses units between IoT devices and cloud data centres. This model utilises three layers: the IoT devices themselves, the fog layer that processes data closer to the source, and the cloud layer that manages storage and more complex computations [2]–[4]. Even though the fog is regarded as a decentralized proximity solution, it presents the computing services to its clients at a distributed level. However, the advent of exponential growth of IoT and mobile device-based applications also leads to challenges in improving QoS such as latency energy consumption and unnecessary bandwidth utilisation [5].

The main contributions of this study are summarised as follows: In this research, we propose an innovative GGCN-based resource scheduler designed to address resource scheduling in fog networks to minimise the average loop delay and total network usage while adhering to constraints such as the number of active sensors and fog nodes in the network. We compared the performance of our proposed GGCN-based resource scheduler with the current state-of-the-art benchmark schemes, including PSO, SJF, and FCFS. Our results demonstrate the superiority of the GGCN in terms of performance and resource allocation efficiency. The proposed 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.

II. Related work

Pham and Huh [6] addressed scheduling problems in a cloud-fog computing system using list scheduling. They proposed a heuristic-based algorithm designed to balance the execution speed (makespan) and the cost of cloud resources. Their focus was on classifying tasks based on their priority and selecting the most suitable nodes for execution, aiming for optimal resource allocation. Simulation results demonstrated a trade-off between the cost of cloud resources and the makespan of executing the workflows. Bittencourt [7] explored the scheduling difficulty for movement applications in the fog and cloud context hierarchy. This method offers the advantage of enhanced mobility and lower latency, but it has the downside of increased temporal complexity. This study, building upon Bittencourt's work, focused on analysing the scheduling issues in fog computing, considering factors such as user mobility, application performance, and three distinct scheduling policies (concurrent, delay-priority and FCFS). The objective was to improve execution efficiency based on application characteristics. The conclusion highlights that fog computing offers a promising approach to addressing resource allocation and management challenges in distributed infrastructures at the network edges.

Yang et al. [8] presented an analytical framework and a novel algorithm called Delay Energy Balanced Tasking Scheduling (DEBTS), for task scheduling in homogeneous fog networks. The primary aim of their study was to strike a balance between service delay and energy consumption in such networks, particularly for delay-sensitive and energy-constrained applications. Their results demonstrated that DEBTS significantly outperforms traditional random scheduling and least busy scheduling algorithms in both energy efficiency and service delay reduction. Wan et al. proposed a unique job scheduling strategy [9] for optimising fog node resource use. The strength of their strategy lies in reducing task delays and enhancing task concurrency, albeit disregarding the constraints of finiteness and information processing time. [10] focused on distributing QoS-aware scheduling for streaming data software applications in a fog networking environment. This study demonstrated that it is possible to increase run-time scheduling while simultaneously decreasing the delay and performance levels. The limitation of the above-mentioned study is a lack of availability and scalability.

A recent study introduced a new fog task scheduler known as FPFTS, which employs particle swarm optimisation and fuzzy theory to enhance resource management in the context of IoT. In an IoT scenario characterised by varying user mobility, fog-device link bandwidth, and latency, the approach of FPFTS significantly reduces both application loop delay and network utilisation. It outperforms traditional FCFS and delay-priority algorithms. Specifically designed to boost performance for latency-sensitive applications, the scheduler extends the cloud computing paradigm to the network edge. This effectively addresses the constraints of resource-limited devices in the IoT environment [11]. Malathy and Revathi [12] proposed a task-scheduling algorithm for fog computing called ECOPRAS. This algorithm combines the entropy weight method and the complex proportional assessment (COPRAS) approach and works in two phases. The first phase involves task prioritisation using a heuristic method known as B-level, while the second phase assigns tasks based on the ECOPRAS method's ranking. ECOPRAS aims to improve the reliability and utilization rate of IoT applications while reducing energy consumption, cost, and makespan. Simulation results show ECOPRAS outperforms comparable algorithms across various workflows.

III. Proposed Methodology

This section provides the considered system architecture as illustrated in Figure 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.

The Core layer represents a heterogenous 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 managing the resource requirements and scheduling tasks received from the IoT devices. This also 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 $F = \{F_1, F_2, \ldots, F_{|N|}\}$: where each fog node are assigned

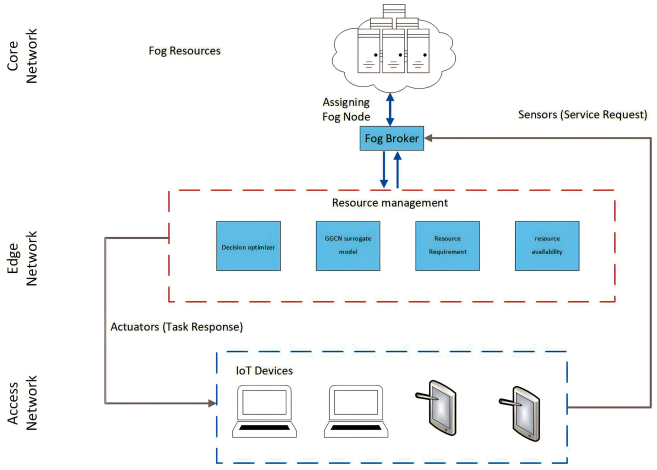

Fig. 1: System Architecture

tasks by the fog broker. Each fog node F_n (where $n \in N$) 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 $|N|$.

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.

A. Problem Formulation

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

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

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.

$$\sum_{n=1}^{N} x_n \leq |M| \tag{2}$$

The number of active fog nodes in the network are constrained such that no more than $|M|$ 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.

$$\sum_{n=1}^{N} y_{n,s} \leq |S_n|, \forall s \tag{3}$$

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

$$\sum_{n=1}^{N} x_n \times y_{n,s} \leq |\mathrm{S}|, \forall s \tag{4}$$

Let D_{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 [13].

$$D_{avg} = \mathrm{CC} - \mathrm{ET}, \tag{5}$$

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

$$U_{\max} = \sum_{n=1}^{N} U_n, \tag{6}$$

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

$$\begin{aligned} \min_{\mathbf{x_n}, \mathbf{y_{n,s}}} \quad & \tfrac{\omega}{D} \times D_{\text{avg}} + \tfrac{(1-\omega)}{U} \times U_{\max} \\ \text{s.t.} \quad & 0 \leq \omega \leq 1 \\ & x_n \in \{0,1\}, \forall n, y_{n,s} \in \{0,1\}, \forall s \\ & (3) - (4) \end{aligned} \tag{7}$$

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.

B. 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 reduce the loop delay and the total network usage. The GGCN is a deep learning-based algorithm that is based on neural networks whose inputs are in the form of graphs. It works by filtering and extracting features from the input data and training those extracted features to create an optimal prediction model.

In GGCN, the graphs can be represented as G= (V, E) where $|v|$ and $|E|$ denote the vertices and edges, respectively. The vertices, also known as nodes, contain unstructured data. The aim of GGCN is to classify nodes into classes by identifying the relationships between them. 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 that share similarities can be clustered together using the neighbourhood aggregation process. During this step, the embedding for a node is the average of the embeddings of all its neighbours passing through the neural network. This process is iteratively performed to cover the entire graph and increase classification accuracy.

The GGCN architecture is a parameter message-passing-based graph neural network that employs residual connections, batch normalisation, and edge gates. The GGCN process involves defining a neighbourhood aggregation function, in which neighbours are assigned weights that determine their impact on node embedding. To implement a GGCN, the first step is to define a neighbourhood aggregation function. In this approach, all nodes are considered to have an equally weighted node, which means that they have the same impact on the generated node embedding. The next step is to define a loss function that is based on the task for which predictions are to be made.

In the GGCN algorithm 1, the inputs (T, V, E) are obtained from the resource manager in the broker. An estimated QoS score (O ˆ) is then generated from the inputs, which is used to generate a scheduling decision. The scheduling decision is made by first splitting the fog devices and tuples into subsets of predetermined size (K). Each tuple is sourced from either

Algorithm 1 GGCN algorithm

Input [T,V,E]
Output Executed Tuple

procedure IFOGSIM
  T,V,E ← ResourceManager()
  O ^ ← f ([T,V,E]) estimate QoS score
  Tuples ← get top K tuples based on deadlines
  Fog Nodes ← get bottom K nodes based Ram and Mips
  for <t ∈ tuples> **do**
    if allocation of t to h is feasible **then**
      Allocate t ← h
    else
      Add t to wait queue.

sensors or fog nodes and is sorted based on its deployment time and deadlines. The fog broker verifies the availability of resources in the fog nodes to process incoming tuples. The fog nodes that have lower resources are assigned short deadline tuples. If all the fog nodes are unavailable, the tuples are placed on a waiting list. The process is then iterated until all the tuples have been assigned and processed.

IV. PERFORMANCE EVALUATION

This section presents the outcomes of the GGCN mechanism in various scenarios in comparison with the state-of-art mechanisms of FCFS, SJF and a modified PSO. We select two performance metrics: average loop delay and total network usage. Moreover, in this section, we focus on our experimental setup, configuration, different scenarios, and results in detail. Figure 2 illustrates the general topology of our system, summarising the number of clouds, gateways, fog nodes, sensors and actuators employed.

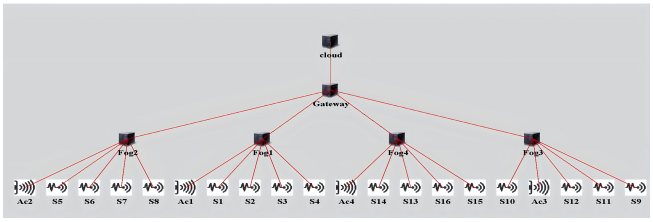

Fig. 2: Fog computing topology designed in Ifogsim

A. Experiment Setup

For the valuation of our proposed deep learning scheduling approach, we developed our scenario and implemented our approaches using extending some packages of Ifogsim. Table I shows the simulation setup for our scenario. Both our scenario 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 I: 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

B. Configuration

Table II illustrates the parameters of fog computing architecture configuration in our scenario, including the cloud fog node and gateway. We consider three layers in our scenario. 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 II: 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

C. Scenario

Our hypothesis scenario is a heterogeneous scenario, 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.

The scenario would be implemented in IfogSim; some of the classes that would be used are introduced below:

Sensors: This would be used to simulate IoT sensors. The sensors would generate tuples that could be transmitted to the fog devices.

Fog device: This class would be used to simulate fog devices. For each fog node, the memory, storage size, processor, uplink, and downlink would be specified. The fog nodes could process the tuples based on scheduling decisions made by the

fog broker.

Tuples: They would represent the packets of information passed across from every one place to another in the IfogSim environment. The tuples would contain information about their source and their destination.

D. Results and Discussion

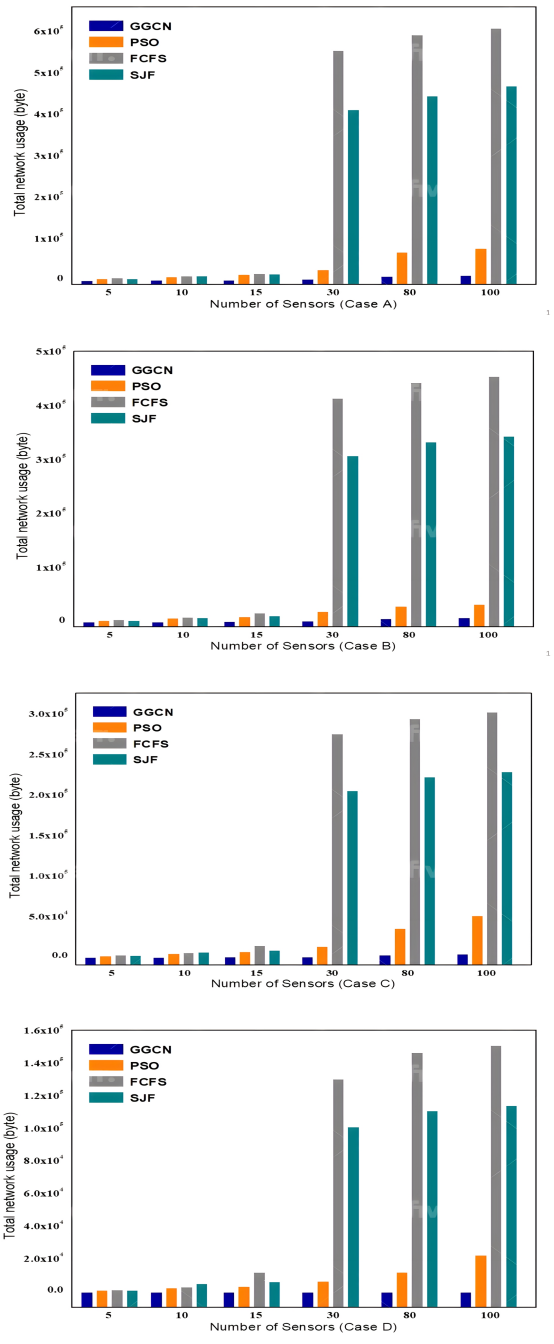

Fig. 3: Total network usage (byte)

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

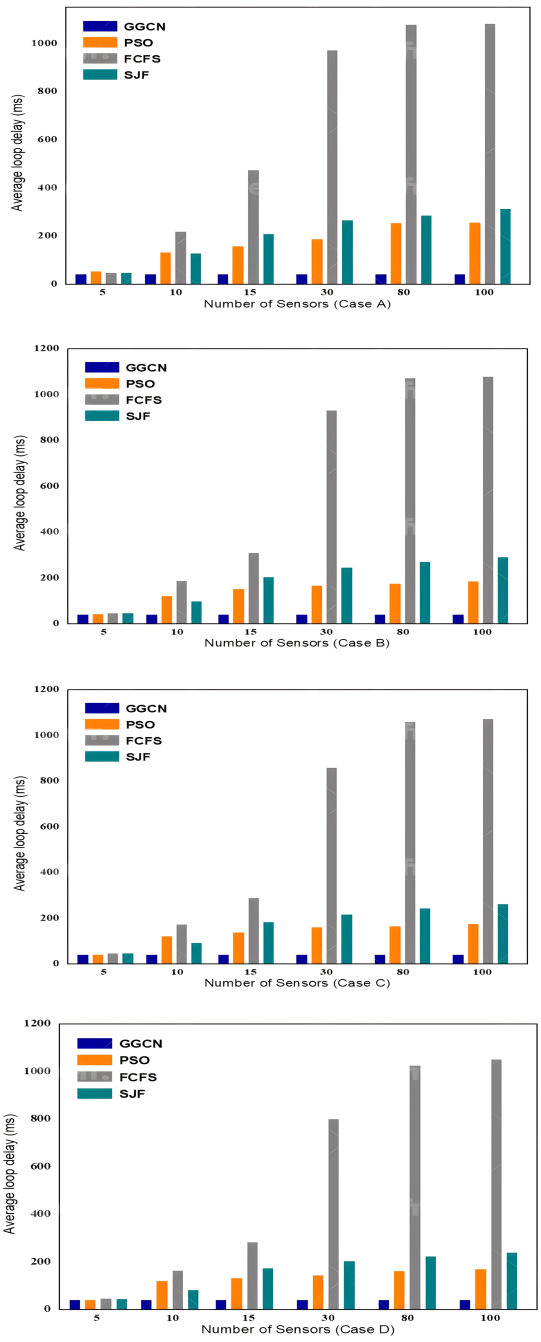

Fig. 4: Average loop delay (ms)

algorithm is compared with benchmark algorithms such as FCFS, SJF and PSO approaches in terms of allocating proper resources to process incoming tasks. Figure 3 illustrates the total network usage in different cases such as A: Four fog nodes; B: Three fog nodes; C: two fog nodes; and D: One fog node. 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 (Figures 3 and 4). Figure 3 demonstrates that four resource scheduling algorithms (GGCN, PSO, FCFS, and SJF) were evaluated in terms of their impact on network usage across different scenarios (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 scenario A with 100 sensors, SJF, PSO, and FCFS yielded 320.2, 262.9, and 1089.23, respectively, which are higher than the GGCN outcome. In the case of A, GGCN resulted in the lowest usage of 1122.9 with only 5 sensors, and even with 100 sensors, the network usage remained significantly lower than the other approaches. In case D, the network usage ranged from 7.92 to 190, 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.

Figure 4 presents an 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 the case of A, GGCN achieved the minimum average loop delay of 47.25 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 A, SJF and PSO resulted in average loop delays of 320.2 and 262.9, respectively, both exceeding the delay produced by the GGCN approach. FCFS too, under identical conditions, demonstrated the highest loop delay of 1089.23. This trend, with GGCN delivering the lowest loop delay and FCFS the highest, persisted across all configurations - Cases B, C, and D. In Case D, GGCN continued to outperform the others, recording the lowest delay of 46.72 with 5 sensors, while the FCFS approach reported the highest delay at 1058.49 with 100 sensors.

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.

V. 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. Our proposed algorithm shows improved results compared to FCFS, SJF and PSO. This study shows 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.

References

[1] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, 'Resource management approaches in fog computing: a comprehensive review', Journal of Grid Computing, vol. 18, no. 1, pp. 1–42, 2020.

[2] V. Kumar, A. A. Laghari, S. Karim, M. Shakir, and A. A. Brohi, 'Comparison of fog computing & cloud computing', Int. J. Math. Sci. Comput, vol. 1, pp. 31–41, 2019.

[3] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, 'A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing', IEEE Access, vol. 7, pp. 115760–115773, 2019.

[4] M. M. Sadeeq, N. M. Abdulkareem, S. R. M. Zeebaree, D. M. Ahmed, A. S. Sami, and R. R. Zebari, 'IoT and Cloud computing issues, challenges and opportunities: A review', Qubahan Academic Journal, vol. 1, no. 2, pp. 1–7, 2021.

[5] M. I. Bala and M. A. Chishti, 'Offloading in cloud and fog hybrid infrastructure using iFogSim', in 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2020, pp. 421–426.

[6] X.-Q. Pham and E.-N. Huh, 'Towards task scheduling in a cloud-fog computing system', in 2016 18th Asia-Pacific network operations and management symposium (APNOMS), 2016, pp. 1–4.

[7] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, 'Mobility-aware application scheduling in fog computing', IEEE Cloud Computing, vol. 4, no. 2, pp. 26–35, 2017.

[8] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, 'DEBTS: Delay energy balanced task scheduling in homogeneous fog networks', IEEE Internet of Things Journal, vol. 5, no. 3, pp. 2094–2106, 2018.

[9] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, 'Fog computing for energy-aware load balancing and scheduling in smart factory', IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4548–4556, 2018.

[10] V. Cardellini, V. Grassi, F. L. Presti, and M. Nardelli, 'On QoS-aware scheduling of data stream applications over fog computing infrastructures', in 2015 IEEE Symposium on Computers and Communication (ISCC), 2015, pp. 271–276.

[11] S. Javanmardi, M. Shojafar, V. Persico, and A. Pescapè, 'FPFTS: A joint fuzzy particle swarm optimization mobility-aware approach to fog task scheduling algorithm for Internet of Things devices', Software: practice and experience, vol. 51, no. 12, pp. 2519–2539, 2021.

[12] N. K. Malathy and T. Revathi, 'Entropy-based complex proportional assessment for efficient task scheduling in fog computing', Transactions on Emerging Telecommunications Technologies, vol. 34, no. 2, p. e4690, 2023.

[13] A. Sallam et al., 'Performance Evaluation of Fog-Computing Based on IoT Healthcare Application', in 2021 International Conference of Technology, Science and Administration (ICTSA), 2021, pp. 1–6.