

Fog-Computing Multi-Metric Hybrid Dynamic Programming Latency and Energy Optimization

Nengak I. Sitlong^{1*}, Abraham E. Evwiekpaefe², Martins E. Irhebhude³

Department of Computer, Faculty of Science, Federal University of Education (FUEP), Pankshin, Plateau State, Nigeria¹

Department of Computer, Faculty of Military Science and Interdisciplinary Studies, Nigerian Defence Academy (NDA), Kaduna, Nigeria^{2,3}

iliya_sitlong@yahoo.com¹, aeevwiekpaefe@nda.edu.ng², mirhebhude@nda.edu.ng³

Article Info

Article history:

Received Oct 03, 2025

Revised Oct 24, 2025

Accepted Jun 2, 2026

Keyword:

Fog Computing
Task Scheduling
Response Time
Energy Efficiency
Optimization

ABSTRACT

Efficient task scheduling in fog computing requires balancing multiple quality of service (QoS) parameters, including response time, Service Level Agreement (SLA) compliance, container reliability, migration overhead, and energy consumption. This study evaluated seven optimization algorithms; hybrid DP+LSTM, GOBI2, DRL, GOBI, POND, MILP, and GA based on their impact on average response time while considering migration cost, container destruction, SLA violations, Wait time and energy per container. Results reveal that DP+LSTM achieves the lowest average response time (82.91 ms), outperforming GOBI2 (395.13 ms) by 79.0%, DRL (512.41 ms) by 83.8%, GOBI (776.09 ms) by 89.3%, POND (2789.09 ms) by 97.0%, MILP (35,720.49 ms) by 99.8%, and Genetic Algorithm (145,465.53 ms) by 99.94%. Additionally, Hybrid DP+LSTM maintains zero SLA violations, only three destroyed containers, and the lowest migration overhead (0.018 ms), while also consuming the least energy per container (2,835,048 J). In contrast, MILP and GA incur extreme delays, high container destruction (351 and 449, respectively), hundreds of SLA violations (397 and 223, respectively), thousands of milliseconds of wait time (11,835ms and 5,251ms, respectively) rendering them unsuitable for practical fog environments. Overall, hybrid DP+LSTM demonstrates superior scalability, energy efficiency, and QoS preservation, making it the most effective scheduling strategy for fog computing environments.

© This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Corresponding Author:

Nengak I. Sitlong

Department of Computer, Faculty of Science, Federal University of Education (FUEP)

Pankshin, Plateau State, Nigeria

Email: iliya_sitlong@yahoo.com

1. INTRODUCTION

Fog computing extends cloud capabilities closer to end-users by deploying computation, storage, and networking resources at the network edge. This architectural shift is intended to reduce service latency and improve energy efficiency for delay-sensitive and resource-intensive Internet of Things (IoT) applications [1]. However, the performance of fog environments is not determined solely by hardware placement, it is shaped by several interrelated operational metrics. These

metrics includes Service Level Agreement (SLA) violations, average energy per container destruction, energy per container interval, wait time, average migration time, number of migrations, and number of containers destroyed [2]. Each of these metrics influence both response time and energy consumption in measurable ways.

SLA violations are a direct indicator of unmet QoS targets. In fog computing, violations often arise from excessive processing delay, network congestion, or resource contention [3]. Each violation represents a breakdown in latency guarantees, and sustained high violation rates typically imply inefficient resource allocation or migration strategies that also waste energy through unnecessary re-executions or retries [4]. Average energy per container destruction captures the one-time energy overhead of terminating a container and initializing its replacement [5]. Frequent or poorly timed destructions lead to energy spikes and degraded responsiveness due to cold-start delays [6]. Energy per container interval refers to the average operational energy over a defined time slice that offers complementary view, revealing baseline consumption patterns and the efficiency of consolidation or scaling strategies under stable loads [7].

Wait time (queueing delay) directly affects user-perceived response time. In fog environments, where devices are heterogeneous and resources limited, queue buildup can arise from overwhelming workloads or suboptimal load balancing [3]. Longer waits not only degrade QoS but can increase energy usage if workloads are forced to migrate or idle containers remain active awaiting execution. Average migration time refers to the duration to move a container or VM between nodes. This effect can be a significant contributor to end-to-end latency, especially for stateful services [8]. The number of migrations reflects how frequently such transfers occur. While migration can be beneficial for load balancing or mobility support, excessive migration rates amplify energy consumption due to data transfer, memory checkpoint, and potential downtime, thereby also impacting SLA compliance [9]. Finally, the number of containers destroyed measures the rate at which workloads are forcibly terminated rather than scaled down or suspended. High destruction rates often indicate reactive, rather than predictive, resource management, leading to avoidable energy costs and service delays [10].

Research has progressively shifted from heuristic and metaheuristic scheduling to machine learning and deep reinforcement learning (DRL) approaches that jointly optimize these metrics [1]. Multi-objective frameworks now routinely consider SLA adherence, migration overhead, and container lifecycle management alongside traditional goals like energy minimization [4]. This evolution reflects a growing recognition that response time and energy efficiency in fog computing are emergent properties of the interplay between these operational metrics. Effective optimization therefore demands coordinated control, where reducing one metric (e.g., migration time) does not inadvertently worsen another (e.g., energy per container destruction). Understanding and managing SLA violations, container lifecycle energy costs, wait times, and migration behaviors is central to achieving the dual objectives of low latency and high energy efficiency in fog computing [2]. These metrics not only provide diagnostic insight into system health but also serve as levers for advanced scheduling algorithms that can adapt to the dynamic and heterogeneous nature of edge-to-cloud ecosystems.

Gamut of techniques have been deployed by researchers in managing SLA violations, container lifecycle energy costs, wait times, and migration behaviors in edge-cloud device latency and energy optimization. Some of these techniques with their advantages and drawbacks are discussed in this section.

1.1 Heuristics Techniques

The focus in heuristic techniques is to handle classic service placement and energy-aware scheduling. Heuristics algorithms such as Integer Linear Programming (ILP) and meta-heuristics such as Genetic Algorithm (GA) adapted from cloud to fog; early container scheduling assessments. The effects of these techniques on metrics that affects IoT cloud response time and energy consumption are enormous. Priority network and energy-aware placement was used to reduced path length and congestion cutting down response time and SLA breaches compared with naive placement approach by 18% and 14% increase in energy consumption and response time,

respectively [15]. Energy per interval and container, otherwise referred to as energy minimized via consolidation and turning off idle nodes was implemented, however, the spikes outcome during reconfigurations were not always modeled causing interval energy variation under load bursts [16]. Container migration and destruction handling was promising in a container-first scheduling techniques deployed in cloud-edge setting, a multi-objective placement for latency + energy + reliability; comprehensive fog resource-allocation surveys.

Container-based task scheduling and improved placement consistently lowered end-to-end latency and SLA violations compared to VM-only baselines techniques. This was demonstrated using reinforcement-learning (RL) and meta-heuristics (improved JAYA) which showed strong trade-off control between metrics [17]. Better co-location and dynamic scaling reduced average energy per interval while maintaining queueing delay bounds [18]. The research showed that container-migration models highlighted latency jitter and task offloading-overhead and empirical tuning reduced average migration time but frequent mobility still inflated number of migrations.

1.2 Learning-driven Techniques

Deep Reinforcement Learning (DRL) for micro service/container scheduling, auto-encoder-assisted consolidation, carbon/thermal awareness, and migration-centric control to keep QoS while cutting energy was implemented [19]. DRL schedulers (and hybrid ML approaches) learn workload/mobility patterns, cutting down latency and SLA breaches compared with heuristic baselines in heterogeneous fog/edge [20]. Consolidation (VM + containers) and energy-aware scheduling reduce node-hours and cooling/thermal costs several studies explicitly report energy impact of leaving/terminating containers, making “per-destruction” energy overhead a tracked quantity [21]. Learning-based placement/routing lowers queueing time by steering jobs away from impending hotspots, container cold-start and I/O bottlenecks that, if ignored, re-inflate waits [22]. Industrial-edge research modeled migration as a first-class control for event-triggered or prediction-based migrations system that reduce migration count and amortize the time cost by using pre-copy and delta streaming techniques to improve perceived responsiveness for IoT device users [23]. Modern policies increasingly separate “scale-to-zero” from eviction/destruction and account for the one-time energy and SLA risk of teardown warm-start. The tuned policies reduce unnecessary terminations during brief dips, curbing container destroyed counts and associated energy spikes [21]. However, nonlocal jumps often cause excessive migrations. This was mitigated by Predictive Offloading for Network Devices (POND), a Max-Weight variant [24].

1.3 Hybrid Techniques

A hybrid, multi-objective task scheduling method combining the Artificial Bee Colony (ABC) algorithm with a Q-learning algorithm [25] has been introduced as an independent scheduling strategy for cloud computing. This method aims to address simultaneous challenges by optimizing both scheduling and resource allocation, improving virtual machine (VM) throughput, and ensuring load balancing among VMs based on makespan, cost, and resource usage. Its performance was evaluated using three datasets Random, Google Cloud Jobs (GoCJ), and Synthetic workload and compared against existing load balancing and scheduling techniques. Results showed that the proposed method surpasses existing approaches in terms of resource utilization, cost efficiency, and makespan reduction [26].

Similarly, another hybrid solution called the Influential Load Balancing Scheme (ELBS) was implemented [27], designed for healthcare applications. This approach achieves load balancing in fog-cloud environments by integrating a real-time scheduling algorithm with a caching mechanism. The methodology consists of five modules that ensure stable interconnections among fog nodes. Developed and tested using iFogSim, ELBS was compared with other load balancing methods to evaluate its effectiveness.

Enhanced Gradient Based Optimization with Back Propagation to Input (GOBI2) algorithm [11], produced one of the best IoT cloud QoS with improved energy consumption and reduced latency response time. GOBI2 achieved this by feeding back the same result from a surrogate neural network model back to itself through a Monte Carlo simulation to eliminate local optima problem. This process converged after multiple undefined number of iterations because the

confident level of the best response time was unknown which resulted into neural network saturation, a situation where feeding more data to machine learning does not necessarily improve its accuracy but add computational complexity [12]. Furthermore, the optimized latency obtained using Mixed Integer Linear Programming (MILP) model was 73% as compared to cloud only IoT response time [13]. In addition, the standard medically guaranteed safety response time margin is set at 150 to 400 milliseconds [14]. Hence, this research intends to enhance the quality of service, specifically the energy consumption and response time of fog computing system, to surpass the current record through the use of a hybrid model that combines heuristic technique, precisely dynamic programming technique, and Long Short-Term Memory (LSTM) technique.

1.4 Limitations on the Realism of the Research Result

The research result relies completely on the object-oriented model's standard defined by the simulation platform used, precisely the Couple Simulation with Container Orchestration (COSCO), couple here implies the platform is capable of handling both simulated and actual cloud operations. The platform has objects defined for models such as power, response time, wait time, container destruction, Service Level Agreement (SLA). The SLA violation deadlines are set based on the parameters in equation (8) whose returned value depends on the number exceeding the permissible number of containers destroyed when task migration occurs due to non optimal task allocation by an algorithm. Hence, the possibility of having a zero SLA violation suggesting optimal task allocation and user experience is possible if the number of containers destroyed does not exceed the allowable range as set by the simulator. Therefore, despite the data used for the analysis is actual cloud traces from bitbrain database, there is still room for limitations in result generated due to the simulated platform set standards.

2. PROPOSED METHOD

The proposed framework integrates a Hybrid Dynamic Programming (DP) and Long Short-Term Memory (LSTM) model as shown in Figure 1, to optimize latency and energy consumption in containerized fog computing environments. The system begins with workload traces and fog node states obtained from bitbrains datasets, a repository of workloads and states from a real-life datacenter. The DP component applies a model-based optimization strategy to generate candidate container/task placement and migration decisions using equations (1) and (2), while also producing intermediate cost structures such as energy and response time values equations (12) and (13), respectively. These outputs are not only used for immediate decision-making but are also passed as contextual features into the LSTM predictor. The LSTM network is a pre-trained model that leverages sequential workload history together with DP-derived features to forecast multiple quality-of-service (QoS) and resource-related metrics, including the summations of destroyed containers equation (3), migrations equation (4), total energy equation (5), average response time equation (13), migration time equation (7), SLA violations equation (8), SLA violation percentage (9), waiting time equation (10), energy per container interval equation (6), as well as average energy consumption equation (12). These predicted values are aggregated through a multi-objective function equation (11), balancing trade-offs between latency, energy efficiency, and SLA adherence.

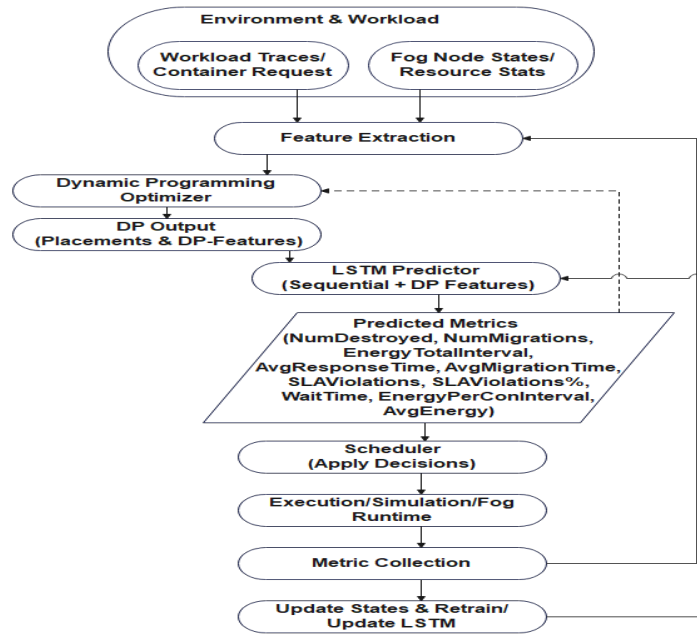


Figure 1. Proposed Model

The incoming workload and fog node states from the bitbrains repository are firstly pre-processed through feature extraction module Figure 2. Final scheduling decisions are executed in the fog runtime, where actual metrics are collected and compared against predictions. The feedback loop continuously updates both the feature extraction process and the LSTM model, enabling online adaptation to workload dynamics. By combining DP's structured optimization with LSTM's sequential prediction capabilities, the framework achieves improved decision accuracy, robustness to dynamic workloads, and holistic multi-metric optimization in fog computing environments. In summary when a job or task j is assigned to a machine VM_j , and the required resource utilization level for carrying out the job are estimated, that is its Instruction Per Second (IPS), RAM, Disk, and Bandwidth requirements, in this case estimated as c , will be compared against the available resources of the machine, in this case given as w_j , to determine the optimal profit p_j that will be derived from the task execution process on the selected machine.

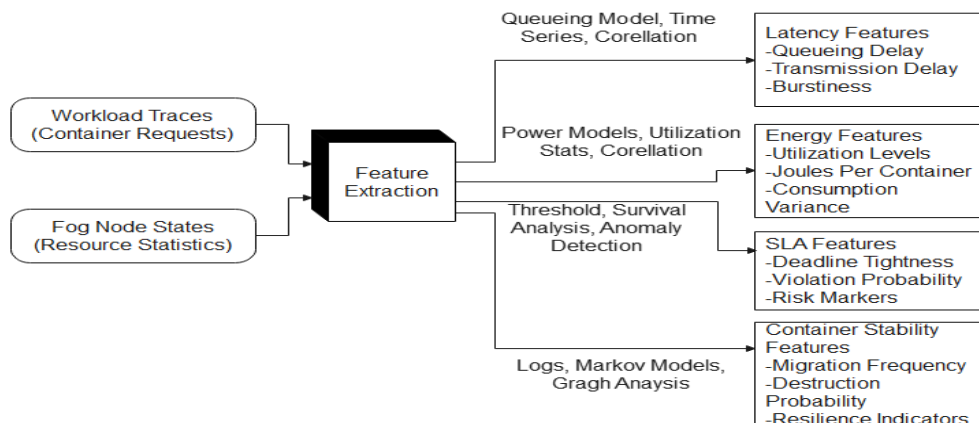


Figure 2. Feature Extraction

In modelling the dynamic programming optimization problem, this research presents a generalization of the illustrated approach referred to as the 0-1 Dynamic Programming problem with multiple containers [28]. In this case, the multiple containers are incoming jobs or tasks from

the IoT layer (bitbrains) dataset to the dynamic programming scheduler in Figure 1. Hypothetically, there will be m tasks with respective capacities of w_i , ($i = 1, \dots, m$). Therefore, the binary vector was represented as x_{ij} , j being the VM in which the task i is allocated. Hence, the optimal profit will be p_{ij} , the required resources will be w_{ij} , and the virtual machine available resources will be c_{ij} . This situation is expected to maximize the QoS function D_3 subject to the constraint D_4 as specified.

$$D_3 = \sum_{i=1}^m \sum_{j=1}^n p_{ij} x_{ij} \quad (1)$$

$$D_4 = \sum_{i=1}^m \sum_{j=1}^n w_{ij} x_{ij} \leq c_i, \quad i = 1, \dots, m \quad (2)$$

Subject to:

$$\begin{aligned} \sum_{i=1}^m x_{ij} &\leq 1, & j &= 1, \dots, n \\ x_{ij} &= 0 \text{ or } 1, & i &= 1, \dots, m, j = 1, \dots, n \end{aligned}$$

2.1 Multi Metric QoS Models:

To understand the response time and energy consumption models, other key metrics computation models such as number of containers destroyed, migrations, Service Level Violations (SLA) and wait time needs to be explained clearly.

Number destroyed refers to number of containers destroyed in an interval computed using equation (3).

$$numdestroyed_t = |destroyed_t| \quad (3)$$

Migration occurs when task are destroyed and reinitiated, number of container migrations in an interval is computed as in equation (4).

$$nummigrations_t = |migrations_t| \quad (4)$$

Energy consumption per interval refers to total host energy consumed in an interval as given in equation (5).

$$energytotal\ interval_t = \sum_{h=1}^H P_h(t) \cdot \Delta t \quad (5)$$

Where:

$P_h(t)$ = Host h 's instantaneous power

Δt = Time interval

Energy per container interval refers to the energy consumed per active container in an interval given as equation (6)

$$energypercontainer\ interval_t = \frac{energytotal\ interval_t}{N_{active}(t)} \quad (6)$$

Where:

$N_{active}(t)$ = Number of active containers at time interval t .

Average migration time refers to the average migration time of destroyed containers in an interval computed presented as equation (7).

$$avemigrationtime_t = \begin{cases} \frac{1}{|destroyed_t|} \sum_{c \in destroyed_t} T_c^{mig}, & |destroyed_t| > 0 \\ 0 & otherwise \end{cases} \quad (7)$$

Where:

C = Container destroyed

T_c^{mig} = Migration time per container

Service Level Agreement (SLA) violation refers to number of destroyed containers exceeding their SLA deadline in an interval as represented in equation (8).

$$slaviolationst_t = |\{c \in destroyed_t | c.destroyAt > SLA_t\}| \quad (8)$$

Where:

$c.destroyAt$ = Number of containers destroyed at time interval t

SLA_t = Acceptable SLA deadline at time t .

SLA violations percentage refers to the percentage of destroyed containers violating SLA in an interval computed as shown in equation (9).

$$slaviolationpercentage_t = \begin{cases} \frac{slaviolationst_t}{|destroyed_t|} \times 100, & |destroyed_t| > 0 \\ 0 & otherwise \end{cases} \quad (9)$$

Wait time refers to the time between creation and start of each destroyed container in an interval as shown in equation (10).

$$waittime_t = \{(c.startAt - c.createdAt) | c \in destroyed_t\} \quad (10)$$

Where:

$c \in$ Container destroyed

$startAt$ = Time the task starts processing

$createdAt$ = Time the task was created

The cloud-fog broker carries out task migration scheduling to ensure tasks are properly assigned to the suitable Virtual Machines. This scheduling process aims to optimize essential IoT Quality of Service (QoS) metrics, specifically response time and energy usage. The objective function is therefore defined based on achieving optimal response time and energy consumption within the given time interval of T_k as $O(P_k)$ equation (11).

Such that:

$$O(P_k) = \alpha.AEC_k + \beta.ART_k \quad (11)$$

Where:

$AEC = \text{Average Energy Consumption}$

$ART = \text{Average Response Time}$

$AEC \ \& \ ART \in P_k$

The average energy consumption is computed for any time interval for a device normalized by the maximum power of the device as in equation (12).

$$AEC_k = \frac{\sum_{vm_k \in VM} \int_{k=s(T_k)}^{s(T_{k+1})} Power_{vm_k}(T) dT}{|W_k| \sum_{vm_k \in VM} Power_{vm_k}^{\max} * (T_{k+1} - T_k)} \quad (12)$$

Where:

$W_k = \text{Set of active tasks at time interval } T_k$

$VM_k = \text{Total Number of hosts at interval } T_k$

$dT = \text{uniform time scheduling intervals of duration}$

$T_k = K\text{-th interval}$

$s(T_k) = \text{start point, such that } s(T_0) = 0, \text{ and } s(T_k) = s(T_{k-1}) + d \text{ for every } K > 0.$

The average response time is computed for the interval T_k for all completed tasks C_k normalized by the maximum response time before the present interval as shown in equation (13).

$$ART_k = \frac{\sum_{T_j^k \in T_k} \text{Response Time}(T_j^k)}{|C_k| \max_s \leq k \max_{T_j^s \in C_s} \text{Response Time}(T_j^s)} \quad (13)$$

2.1 Long Short-Term Memory (LSTM) Model:

The core principle of LSTM lies in its memory cells and gates, which regulate what information is retained at any given moment [29]. The LSTM computation is expressed thus; let x_t, h_t and y_t denote the input feature vector, hidden states and output vector, respectively. Given the input sequence $(x_1, x_2, \dots, x_T)$ supplied to the LSTM, a corresponding $(h_1, h_2, \dots, h_T)$ and $(y_1, y_2, \dots, y_T)$ are generated for iterating through equation (14) and (15) for values of $t = 1, \dots, T$.

$$h_t = f_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (14)$$

$$y_t = W_{hy}h_t + b_y \quad (15)$$

Where:

W_{xh} = Input weight matrix, W_{hh} = Recurrent weight matrix, and W_{hy} = Output weight matrix, b_h and b_y denotes the bias terms, and $f_h()$ denotes the activation function. LSTM implemented the activation function $f_h()$ thus:

$$i_t = \sigma_{\logistic}(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (16)$$

$$f_t = \sigma_{\logistic}(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (17)$$

$$\tilde{c}_t = \sigma_{\tanh}(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (18)$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \quad (19)$$

$$O_t = \sigma_{\text{logistic}}(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (20)$$

$$h_t = O_t \odot \sigma_{\tanh}(c_t) \quad (21)$$

Where:

$\odot$ depicts element-wise multiplication, W_{xi}, W_{xf}, W_{xc} and W_{xo} denotes $N \times M$ weight matrices, a W_{hi}, W_{hf}, W_{hc} and W_{ho} denote $N \times N$ weight matrices, b_i, b_f, b_c , and b_o denote $N \times 1$ bias terms, and i_t, f_t, c_t , and o_t are $N \times 1$ vectors called the state of input gates, forget gate, memory cell and output gate, respectively. With value of h_t derived from equation (21), the output results are generated using equation (15). Therefore, considering equations (16 to 21) a total of $4(NM + N^2 + N)$ parameters are learned in a single LSTM layer.

In Figure 1, the workload traces and fog node states are treated as a finite set of tasks with an upper limit M , thus, at any interval k the workload trace W will be $|W_k| \leq M$. The fog node states are expressed as a utilization matrix capturing CPU, RAM, bandwidth, and disk usage, represented as a potential vector of size Z . Consequently, the task utilization matrix $\{U(w_j^{k-1}) | \forall j, w_j^{k-1} \in W_{k-1}\}$ becomes an $M \times Z$ matrix represented as $\phi(W_{k-1})$ represented in equation (14) where the upper rows correspond to active tasks W_{k-1} arranged incrementally by their creation intervals, generated through the dynamic programming scheduler in Figure 1. Similarly, another potential vector represents the utilization matrix of virtual machines, which includes CPU, RAM, bandwidth, disk capacities, and latency, VM with a defined size of Z' . Therefore, at each interval T_k for $|VM|$ virtual machines a matrix of $Z' \times |VM|$ is formed denoted as $\Phi(VM_{k-1})$ also captured in equation (22) considering virtual machine utilization matrix of interval T_{k-1} . Eventually, the input to the LSTM model is formed with the set of new task N_k , tasks in waiting queue Q_{k-1} , active tasks W_{k-1} , the completed tasks C_k and the set of decision matrix $|VM| \times M$ denoted as $\Phi(D)$ captured in equation (14) for the training. The model is fed with the input $[\Phi(W_{k-1}), \Phi(VM_{k-1}), \Phi(D)]$ to estimate the QoS value $O(P_k)$ equation (22).

The scenario is modeled as a continuous function $f(x: \theta)$ as an LSTM approximate of $O(P_k)$ with the θ vector as the dependent parameter of the model and x as discrete or continuous parameter of a defined boundary. Specifically, x is made up of the set of utilization matrix of task and virtual machines together with decision matrix. The parameter θ is learn using the input dataset in equations (1) and (2) such that certain loss is optimized for the dataset.

$$x = \{LSTM[\Phi(W_{k-1}), \Phi(VM_{k-1}), \Phi(D)], O(P_k)\}_k \quad (22)$$

The generated traces are saved for training the model, after the training the model parameter θ is found such that the model can estimate a generic function with a degree of accuracy from a large parameter set θ . The problem is then formulated by finding the decision matrix $\Phi(D)$ that will minimize $f(x: \theta)$ as equation (23).

$$\Phi(D) \text{ minimize } \rightarrow f(x; \theta), \forall D \quad (23)$$

$$\forall k \text{ satisfying equation (22)}$$

2.2 Mean Square Error Loss Function:

The LSTM model applied in this study is pre-trained, and its predictive accuracy is evaluated based on the loss values obtained during execution. These loss values indicate the degree of difference between the predicted outputs and the actual ground truth. In predicting the optimal metric $O(P_k)$, whether response time or energy consumption, the model relies on the dataset in equation (22), the loss value is tracked through the use of Mean Square Error (MSE) loss function as shown in equation (24)

$$Loss(f(x; \theta), y) = \frac{1}{T} \sum_{t=0}^T (y - f(x; \theta))^2 \quad (24)$$

Where:

$T = \text{Time interval}$

$\theta = \text{LSTM parameter}$

$(x, y) \in \text{equation (22)}$

2.3 Software Used

The experiments in this study were conducted using a Python-based platform known as Couple Simulation and Container Orchestration (COSCO). This platform supports the development of secure, flexible, and scalable cloud-based IoT solutions, applicable to both simulated and real cloud environments. COSCO also enables the design and programming of sensors through Node.js, using a JSON-based Application Programming Interface (API). These sensors can be configured to interact with the cloud backend in this case, InfluxDB and Oracle VM VirtualBox.

3. RESULTS AND ANALYSIS

In the research seven fog computing algorithms were evaluated namely Genetic Algorithm (GA), Mixed Integer Linear Programming (MILP), Predictive Offloading for Network Devices (POND), Gradient-Based Optimization with Back Propagation to Input (GOBI), Deep Reinforcement Learning (DRL), Improved GOBI (GOBI2), and the proposed model hybrid DP+LSTM. The summary of the result generated for the multi metrics response time, energy consumption, number destroyed, number of migrations, migration time, SLA violations, SLA Violation percentage and wait time is as shown in Table 1. Table 2 shows the validation of result of hybrid DP+LSTM over DP-only and LSTM only results.

Table 1. Multi Metric comparison

Author	Algorithm	Energy per Container (J)	Avg. Response Time (ms)	Num. Destroyed	Num. Migrations	Avg. Migration Time (ms)	SLA Violations	Wait Time(ms)
Our model	Hybrid DP+LSTM	2,835,048.21	82.91	3	14	0.0176	0	6
[1]	DRL	2,973,238.56	512.41	1	31	0.3152	0	1
[13]	MILP	3,005,498.05	35,720.49	351	957	105.43	307	11,835
[24]	POND	3,310,966.64	2,789.09	6	119	2.65	2	8

Author	Algorithm	Energy per Container (J)	Avg. Response Time (ms)	Num. Destroyed	Num. Migrations	Avg. Migration Time (ms)	SLA Violations	Wait Time(ms)
[11]	GOBI2	3,664,337.58	395.13	2	52	0.0834	0	2
[11]	GOBI	4,463,010.29	776.09	3	54	0.8490	0	3
[15]	GA	3,959,935.56	145,465.53	449	2,734	638.63	223	5,251

Table 2. Validation of Hybrid DP+LSTM Multi Metric Performance over DP-only and LSTM-only

Author	Algorithm	Energy per Container (J)	Avg. Response Time (ms)	Num. Destroyed	Num. Migrations	Avg. Migration Time (ms)	SLA Violations	Wait Time
Our model	Hybrid DP+LSTM	2,835,048.21	82.91	3	14	0.0176	0	6
[28]	DP-Only	2,640,936.98	156.57	1	4	0.1944	0	1
[29]	LSTM-Only	3,671,515.11	282.41	1	53	0.0730	0	1

3.1 Container Destruction by Task Allocation Algorithms in Fog Computing Environment

An essential performance metric in edge-cloud latency and energy efficiency is the number of containers destroyed during task scheduling as shown in Figure 3. Excessive destruction indicates instability, inefficient resource utilization, or poor fault tolerance.

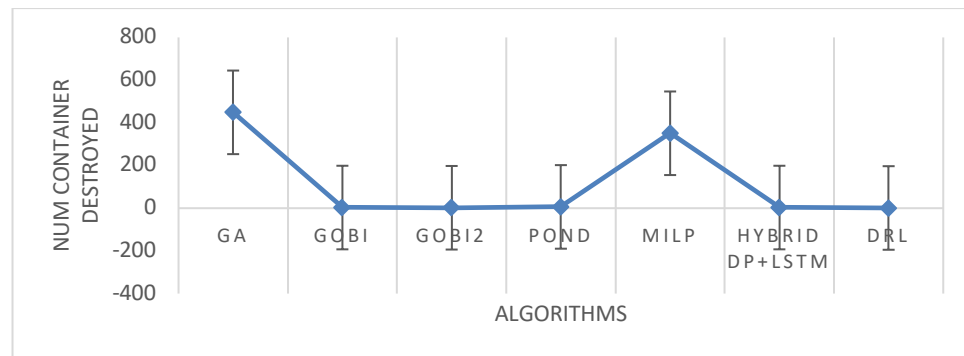


Figure 3. Number of Containers Destroyed

3.2 Number of Migrations by Algorithms

Figure 4 shows number of migrations by algorithms, Genetic Algorithm leads with the highest migrations, followed by MILP. Hybrid DP+LSTM and DRL have significantly fewer migrations, suggesting more stable systems.

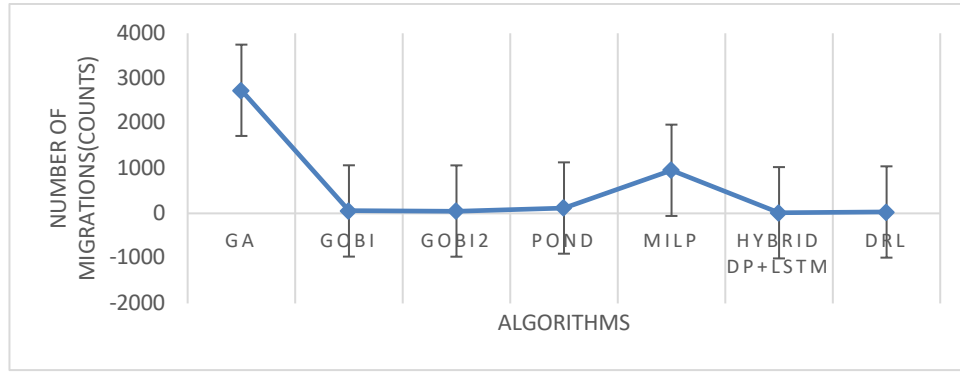


Figure 4. Number of Migrations

3.3 Average Migration Time of Algorithms

Figure 5 shows average migration time with GA having the highest migration time, while Hybrid DP+LSTM and GOBI2 have the lowest, implying faster migrations. Migration time of 0ms indicates zero or negligible task migration overhead.

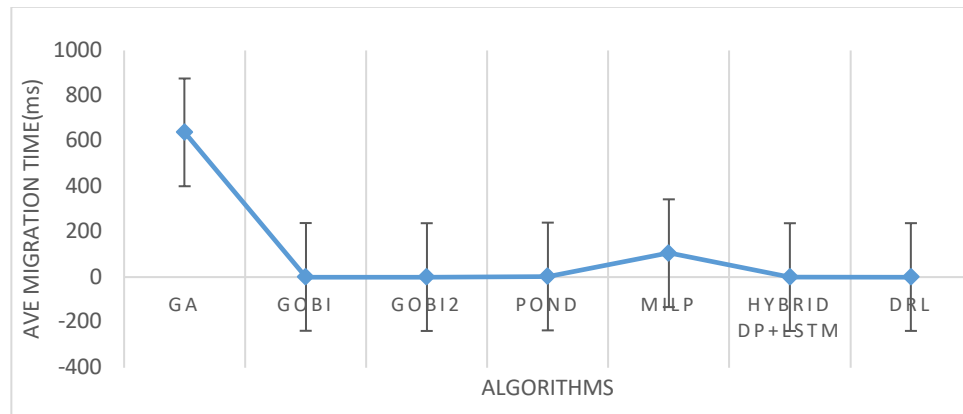


Figure 5. Average Migration Time

3.4 Service Level Agreement (SLA) by Algorithms:

Figure 6 shows Service Level Agreement (SLA) violation. Zero SLA violation value indicates better QoS outcome. These further confirms the ability in offering better task scheduling, load balancing, energy and latency reduction, which are essential in fog computing and key optimization metrics of this research.

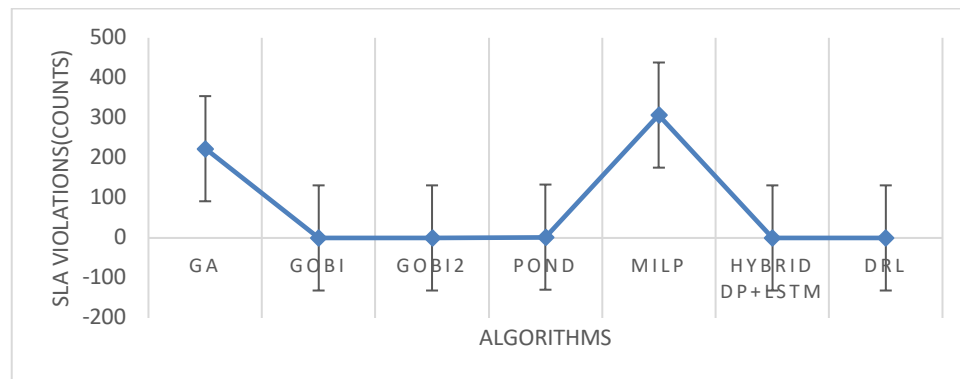


Figure 6. Service Level Agreement (SLA) violations

3.5 Wait Time of Algorithms:

Figure 7 depicts the wait time of algorithms, Lower wait times are directly correlated with faster service delivery, lower latency, and better user experience a crucial requirement in fog computing.

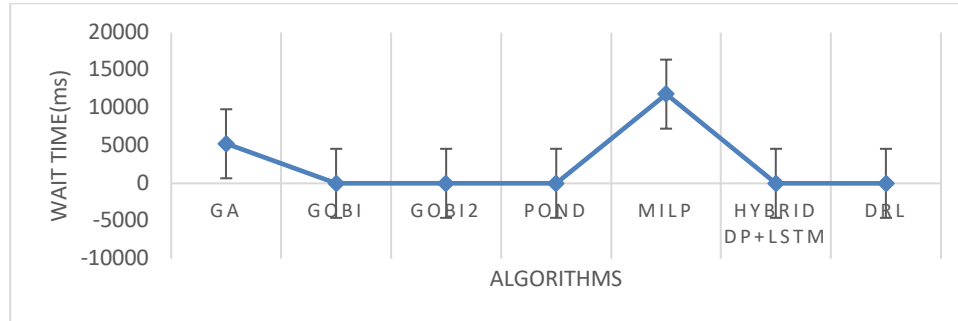


Figure 7. Wait Time

3.6 Algorithm Energy Per-Container Interval:

Figure 8 display the energy per container interval, lower energy per container interval is crucial for extending the lifetime of fog nodes and reducing operational costs.

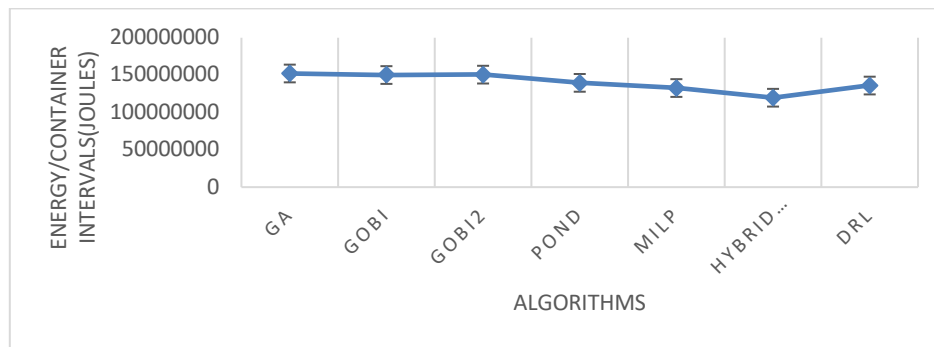


Figure 8. Energy per Container Interval

3.7 Average Energy per Destruction by Algorithms:

Figure 9 presents a comparison of various scheduling/placement algorithms in terms of Average Energy per Destruction (in joules), a term often interpreted in fog/cloud computing as the energy cost associated with completing or migrating (destroying and re-initiating) tasks or containers.

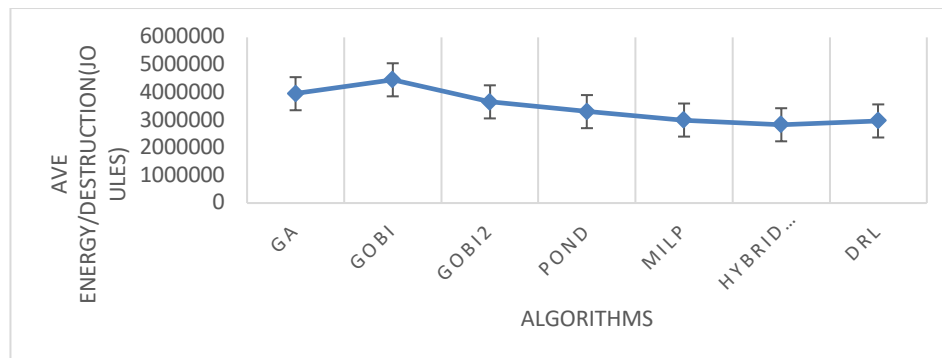


Figure 9. Average Energy per Destruction

3.8 Hybrid DP & LSTM Model Accuracy:

The accuracy of the pre-trained LSTM model was evaluated using the loss values produced by the Mean Square Error (MSE) function. These values indicate the gap between the actual outputs and the model's predictions. As shown in Figure 10, the pre-trained nature of the model explains why the loss began at a low level, decreased slightly during the first epoch, and then stabilized across the subsequent epochs.

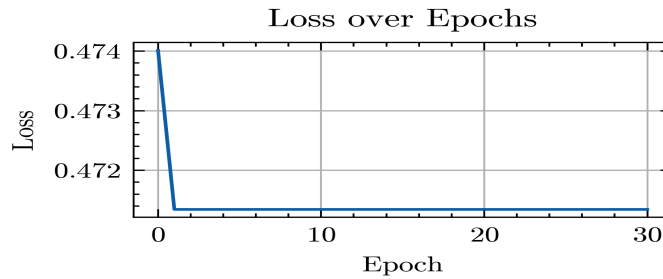


Figure 10. Hybrid DP+LSTM Loss over Epoch

3.9 Analysis of Findings:

From the outcome of the research, the following assertions were deduced:

- a. **Response Time:** The response time was obtained by computing the sum of the solution search time (Scheduling overhead) and the task execution time where Hybrid DP+LSTM achieved the lowest response time (82.91ms), significantly outperforming all other methods with 99.8% over MILP, making it ideal for real-time IoT Cloud applications such as smart healthcare monitoring, autonomous vehicle communication, and industrial automation.

The justifications for this outcome is analysed thus; Table 2 depicts number of containers destroyed, with GA and MILP having the highest values, indicating more frequent destruction of containers, whereas DRL, GOBI, and GOBI2 exhibit minimal destruction. GA (449) and MILP (351) resulted in the highest container destruction. These results suggest that traditional optimization and feedforward neural networks lack the dynamic adaptability required for stable task placement in fog environments. The high destruction count reflects frequent task migration, suboptimal scheduling, or instability in task execution continuity. POND (6) demonstrates more reasonable destruction levels, this is due to its rule-based prioritization mechanism that balances task urgency with node stability. DRL (1), GOBI2 (2), GOBI (3), and Hybrid DP+LSTM (3) showed excellent performances.

Mixed Integer Linear Programming (MILP) shows moderate migration Count (~1100+) hence the moderate potential latency and moderate energy efficiency. This effect emanated from the ability of MILP to formulate task placement as an optimization problem. It guarantees optimal solutions but is computationally expensive, making it slower in dynamic environments. Although accurate, its reaction time leads to more frequent migrations than predictive models. Hybrid DP+LSTM shows extremely low migration Count hence the extremely low latency and high energy efficiency. This hybrid model combines DP with LSTM neural networks to predict future load and make informed decisions. Its predictive nature ensures proactive placements, drastically reducing the need for migrations. This led to the both low latency and high energy efficiency attributed to the hybrid DP+LSTM algorithm. Deep Reinforcement Learning (DRL) reflected a very low migration counts and consequently the low latency and good energy efficiency it provided as shown in Table 1. The effect created by DRL is attributed to its usage of agent-environment interactions to learn optimal placement strategies over time. It adapts to environmental changes dynamically and makes context-aware decisions that reduce

unnecessary migrations. This adaptability results in efficient energy use and low response times.

- b. **Service Level Agreement (SLA) Violations:** DP and LSTM, GOBI, GOBI2, and DRL incurred no SLA violations, making them the most reliable algorithms for IoT service quality assurance.
- c. **Energy Consumption:** Hybrid DP+LSTM had the lowest energy-per-container value (2,835,048 joules), making it energy-efficient and suitable for IoT Cloud sustainability.
- d. **Migration Time:** Hybrid DP+LSTM also achieved the lowest average migration time (0.018ms), highlighting its efficiency in workload distribution across IoT Cloud nodes.

In general, AI/ML algorithms such as DRL, and hybrid models (Hybrid DP+LSTM, GOBI, and GOBI2) are more adaptive to learning optimal scheduling policies to minimize delays dynamically. Traditional optimization models like MILP may not scale or adapt well in the dynamic and distributed fog environment. To achieve high QoS in fog computing Prioritize low wait-time algorithms like DRL, GOBI2, and Hybrid-DP & LSTM and avoid static or computationally heavy algorithms (e.g MILP) for real-time workloads. Low wait time ensures faster task completion, higher user satisfaction, and better resource utilization which are the hallmarks of effective fog computing.

4. CONCLUSION

This study comprehensively evaluates the performance of seven task scheduling and resource allocation algorithms within cloud-based IoT and fog computing environments. The goal was to assess and compare their effectiveness using critical Quality of Service (QoS) metrics such as response time, energy consumption, SLA violations, migration time, wait time, and container destruction. The results confirm that reducing migration delay and SLA breaches are pivotal to achieving low-latency performance, while excessive container destruction significantly undermines efficiency. Among the algorithms GA, MILP, POND, GOBI, GOBI2, DRL, and a hybrid DP+LSTM, hybrid DP+LSTM model consistently outperformed the others across all evaluated dimensions. DP-only with lesser energy consumption is best for energy efficient Industrial purposes whereas LSTM-only lags behind in the key metrics therefore unsuitable for both energy and latency sensitive environments.

REFERENCES

- [1] A. Kallel, M. Rekik and M. A. Khemakhem, "Deep reinforcement learning-based optimization approach for containerized microservice scheduling in hybrid fog/cloud environments," *Engineering Applications of Artificial Intelligence*, vol. 141, no. 2, pp. 109-120, 2024.
- [2] Z. Wang, M. Goudarzi, M. Gong and R. Buyya, "Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments," *arxiv*, vol. 3, no. 1, pp. 26-33, 2023.
- [3] M. Pakmehr, "Delay and energy aware task scheduling mechanism for fog-enabled IoT applications: A reinforcement learning approach," *Computer Networks*, vol. 224, no. 4, pp. 135-140, 2024.
- [4] M. Fahimullah, S. Ahvar, M. Agarwal and M. Trocan, "Machine learning-based solutions for resource management in fog computing," *Multimedia Tools and Applications*, vol. 83, no. 8, p. 23019–23045, 2024.
- [5] Z. Wang, M. Goudarzi and R. Buyya, "ReinFog: A DRL-empowered framework for resource management in edge and cloud computing environments," *arXiv*, vol. 4, no. 2, pp. 43-57, 2024.
- [6] M. P. Toopchinezhad and M. Ahmadi, "Deep reinforcement learning for delay-optimized task offloading in vehicular fog computing," *arXiv*, vol. 4, no. 2, pp. 12-23, 2024.
- [7] W. Shen and M. Tang, "Latency-aware task scheduling for mobile edge computing," *IEEE Transactions on Services Computing*, vol. 15, no. 3, p. 1191–1204, 2022.

- [8] F. U. Khan, I. A. Shah, S. Jan, S. Ahmad and T. Whangbo, "Machine learning-based resource management in fog computing: A systematic literature review," *Sensors*, vol. 25, no. 3, pp. 687-695, 2025.
- [9] A. Singh, A. Kaur and D. Gupta, "Reviewing trust issues in cloud computing," *Journal of Physics*, vol. 1969, no. 1, pp. 12043-12047, 2021.
- [10] S. R. Gunda, "Optimizing Cloud Computing Resource Utilization Through Intelligent Allocation and Containerization Strategies," *Journal of Computer Science and Technology Studies*, vol. 7, no. 8, pp. 238-244, 2025.
- [11] S. Tuli, S. R. Poojara, S. N. Srirama, G. Casale and N. R. Jennings, "COSCO: Container orchestration using co-simulation and gradient based optimization for fog computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 101-116, 2021.
- [12] G. Yan, L. Zhaoze, D. Shuhong, L. Cong, W. Ying, W. Shen, T. Georgios and C. Long, "Deep Reinforcement Learning for Job Scheduling and Resource Management in Cloud Computing: An Algorithmic Level Review," *arXiv*, vol. 1, no. 2, pp. 1-26, 2025.
- [13] H. A. Alharbi, B. A. Yosuf, M. Aldossary and J. Almutairi, "Energy and Latency Optimization in Edge-Fog-Cloud Computing for the Internet of Medical Things," *Computer Systems Science and Engineering, CSSE*, vol. 47, no. 1, pp. 1300-1317, 2023.
- [14] S. Aiswarya, A. Geetha, K. Ramesh, S. Sasikumar and D. Sheema, "Latency Reduction in Medical IoT Using Fuzzy Systems by Enabling Optimized Fog Computing," *International Journal of Electrical and Electronics Engineering*, vol. 9, no. 12, p. 156 – 166, 2022.
- [15] O. H. Hassan, S. Azizi and M. Shojafar, "Priority Network and Energy Aware Placement of IoT-Based Application Services in fog-cloud Environments," *IET Journals*, vol. 14, no. 13, pp. 2117-2129, 2020.
- [16] J. Yang, Z. Saad, J. Wu, X. Niu, H. Leung and S. Drew, "A Survey on Task Scheduling in Carbon-Aware Container Orchestration," *arXiv*, vol. 3, no. 2, pp. 12-23, 2025.
- [17] U. Vadde and V. S. Kompalli, "Energy efficient service placement in fog computing," in *PeerJ Comput Sci*, PMID, 2022.
- [18] J. Bachiega, B. G. Costa, L. R. Carvalho, M. J. Rosa and A. P. Araújo, "Computational resource allocation in fog computing: a comprehensive survey," *ACM Journal*, vol. 55, no. 145, pp. 1-31, 2023.
- [19] A. Khan, F. Ullah and D. Shah, "EcoTaskSched: a hybrid machine learning approach for energy-efficient task scheduling in IoT-based fog-cloud environments," *Sci Rep*, vol. 15, no. 2, pp. 23-36, 2025.
- [20] B. Sreedar, S. P. Goud, K. Yuvraj and G. K. Harin, "A Deep Reinforcement Learning Framework for Optimized Container Scheduling and Load Balancing," *International Research Journal of Multidisciplinary technovation*, vol. 6, no. 6, pp. 4-15, 2024.
- [21] M. Falloon, H. Ma and A. Chen, "Energy-Aware Dynamic Resource Allocation and Container Migration in Cloud Servers: A Co-evolution GPHH Approach," in *Proceedings of the Genetic and Evolutionary Computation Conference*, VIC, Melbourne, Australia, 2024.
- [22] R. Gupta and K. Nahrstedt, "Performance Characterization of Containers in Edge Computing," *arXiv*, vol. 3, no. 2, pp. 40-53, 2025.
- [23] Y. Zhang, K. Zhao, Y. Yang and Z. Zhou, "Real-Time Service Migration in Edge Networks: A Survey," *Journal of Sensor and Actuator Networks*, vol. 14, no. 4, pp. 79-85, 2025.
- [24] X. Liu, B. Li, P. Shi and L. Ying, "POND: Pessimistic-Optimistic online Dispatch," *Journal of arXiv*, vol. 2, no. 1, pp. 34-37, 2020.
- [25] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, no. 3, p. 17803–17818, 2022.

- [26] N. W. Wan, D. Kaharudin, A. J. Muhammad , S. S. Suzi and S. A. Divine, "Survey of Resources Allocation Techniques with a Quality of Service (QoS) Aware in a Fog Computing Environment," *Computers Material and Continua*, vol. 7, no. 1, pp. 34-46, 2023.
- [27] F. M. Talaat, S. H. Ali, A. I. Saleh and H. A. Ali, "Effective load balancing strategy (ELBS) for real-time fog computing environment using fuzzy and probabilistic neural networks," *Journal of Network and Systems Management*, vol. 27, no. 4, p. 883–929, 2019.
- [28] J. V. Guttag, Introduction to computation and programming using Python: with application to computational modeling and understanding data, 3rd ed., Cambride, Massachusetts: Mit Press, 2021.
- [29] B. Sirisha, K. K. Goud and B. S. Rohit, "A Deep Stacked Bidirectional LSTM (SBiLSTM) Model for Petroleum Production Forecasting," in *Procedia Computer Science*, Hyderabad, India, 2023.