Comparative Performance Evaluation of Energy-Efficient and QoS-Aware Fog Scheduling Algorithm

Symah Bashir
School of Computer Applications
Lovely Professional University

Amanpreet Singh
School of Computer Applications
Lovely Professional University

Pankaj Rahi
School of Computer Applications
Lovely Professional University

***Abstract*—Fog computing is crucial for latency-sensitive, distributed applications. Efficient scheduling must balance QoS especially latency with energy constraints on fog nodes. This paper proposes a novel Whale Optimization Algorithm with Convolutional Neural Network (WOAC) energy efficient and QoS aware scheduling algorithm for fog environments. This optimizes for minimizing the task execution delays and reducing overall energy consumption. Through this simulation, WOAC is compared against established baselines using metrics like latency, energy, and deadline violations. Results consistently show WOAC achieving significant reductions in both latency and energy as compared to Artificial Bee Colony (ABC) and First-Come First-Serve (FCFS). It also minimizes deadline violations while maintaining high resource utilization. This proves WOAC effectively balances real-time QoS and energy-efficient in fog computing operations.**

keywords: *Whale Optimization, Convolutional Neural NetworkFog computing, Energy efficient, QoS requirements.*

1.INTRODUCTION

The relentless proliferation of Internet of Things (IoT) devices and the surge in latency sensitive applications ranging from autonomous vehicles and industrial automation to real time healthcare and augmented reality have propelled fog computing to the forefront as a critical complement to cloud computing [1]. By decentralizing computation, storage, and networking resources to the network edge, closer to data sources, fog computing significantly reduces latency, minimizes core network bandwidth consumption, and enhances data privacy and responsiveness. Effective task scheduling the process of mapping diverse computational tasks onto geographically distributed and heterogeneous fog resources is fundamental to realizing these benefits [2]. However, fog environments present unique scheduling challenges due to their inherent resource constraints (limited processing power, energy, especially for battery-powered nodes), heterogeneity (diverse node capabilities), geographical distribution, and dynamic workloads [3]. Energy efficiency and QoS-aware algorithms prioritize meeting stringent application requirements such as low latency, high throughput, reliability, and deadline guarantees [4] [5]. Conversely, energy-efficient algorithms focus on minimizing the power consumption of fog nodes to reduce operational costs, extend hardware lifespan, and promote environmental sustainability [6]. Aggressively pursuing one objective often detrimentally impacts the other minimizing energy via techniques like task consolidation or node sleep states can increase latency, while stringent QoS enforcement may necessitate keeping more nodes active at higher frequencies, consuming excessive energy [7] [8]. Consequently, a multitude of scheduling algorithms have been proposed, employing various strategies (heuristic, meta-heuristic, machine learning-based) and prioritizing different balances between QoS and energy efficiency [9]. Yet, a critical gap exists: the lack of a systematic, comprehensive, and comparative evaluation framework to objectively assess how these diverse algorithms perform across both energy and QoS dimensions under realistic and varied fog scenarios. Understanding their relative strengths, weaknesses, and the specific trade-offs they engender is essential for researchers to advance the state-of-the-art and for practitioners to make informed deployment decisions [10]. This paper addresses this gap by presenting a rigorous Comparative Performance Evaluation of prominent Energy-Efficient and QoS-Aware of scheduling Algorithms. We systematically analyze, implement, and benchmark a representative selection of state-of-the-art algorithms using a comprehensive suite of performance metrics encompassing both QoS parameters (e.g., latency, deadline miss rate) and energy consumption. Through extensive simulations under diverse workload and infrastructure conditions, this study aims to provide crucial insights into the efficacy of different scheduling approaches of energy-QoS trade-offs, and guide the selection and future development of algorithm (WOA) for efficient and sustainable fog computing deployments.

2. RELATED WORK

The critical role of task scheduling in optimizing fog computing performance has spurred significant research, leading to a plethora of algorithms targeting QoS guarantees, energy efficiency, or a balance between both. This section reviews key contributions, categorizing them based on their primary optimization focus and highlighting the existing comparative studies, while establishing the gap this work aims to fill.

2.1 QoS-Aware Scheduling Algorithms

Early and ongoing research heavily focuses on meeting stringent application requirements. Latency/deadline Minimization: Seminal works like [11] emphasized minimizing service latency. Algorithms like Deadline-Aware Dynamic

Task Scheduling (DADTS) [12] and Delay-Priority based methods [13] explicitly prioritize tasks with tight deadlines. Many leverage heuristics (e.g., Earliest Deadline First - EDF variations, Min-Min, Max-Min) or meta-heuristics e.g Genetic Algorithms (GAs), Particle Swarm Optimization (PSO) specifically tuned to reduce response time and Deadline Miss Rates (DMR) [14]. Throughput like approaches will proposed on maximizing system throughput and ensuring fair load distribution across fog nodes to prevent bottlenecks and improve overall responsiveness, indirectly supporting QoS [15].

2.2 Energy-Efficient Scheduling Algorithms

Recognizing the resource constraints of fog nodes, especially battery-powered or remote devices, energy minimization is a major thrust. Resource Consolidation and Dynamic Voltage and Frequency Scaling (DVFS) techniques inspired by cloud computing, such as Virtual Machine (VM) or container consolidation, are adapted to reduce the number of active fog nodes [16]. DVFS is frequently integrated to lower energy consumption of active nodes during computation [17]. Bitam et al. [15] purposes the algorithm for resource scheduling where fog nodes goes into low-power sleep states and wake them up based on demand prediction. Many studies used the optimization for formulate the energy minimization within optimization framework using Linear Programming (LP), meta-heuristics like Ant Colony Optimization (ACO) and Simulated Annealing (SA) algorithms [18].

TABLE I
TAXONOMY OF FOG SCHEDULING ALGORITHM CATEGORIES

Ref	Key Objectives	Common Techniques	Focus
[19]	Minimize Latency/Response Time, Meet Deadlines, Maximize Throughput, Ensure Reliability	EDF variants, Min-Min/Max-Min, GAs, PSO, Deadline-priority heuristics	QoS-Aware
[16]	Minimize Energy Consumption, Maximize Resource Utilization	VM/Container Consolidation, DVFS, Sleep/Wake, ACO, SA	Energy-Efficient
[20]	Optimize Trade-off between QoS Guarantees and Energy Savings	NSGA-II, MOPSO, Weighted Sum, Fuzzy Logic, RL/DNN/Q-learning	Hybrid QoS-Energy

2.3 Limitations of Existing Comparative Studies

the limitations of existing comparative studies on fog computing algorithms. One key limitation is the ”Intra-Category Focus” where comparisons are limited to algorithms within the same primary focus, such as only QoS-aware or only energy-efficient algorithms. This fails to reveal how different algorithm types perform against each other on the energy-QoS trade-off [21]. Another limitation is the ”Narrow Metric Set” where the evaluation focuses predominantly on metrics aligned with one objective, such as latency/DMR or energy, neglecting the other, providing an incomplete picture of overall performance and trade-offs [22]. The ”Limited Algorithm Selection” is another issue, where only a small or non-representative subset of algorithms is evaluated, missing key state-of-the-art approaches, limiting the generalizability of the results [23]. The ”Homogeneous Scenarios” limitation refers to testing under a limited range of workload patterns, network sizes, or resource heterogeneity, failing to demonstrate algorithm robustness for diverse real-world fog environments [24].

Finally, the ”Inconsistent Evaluation Environments” where algorithms are compared using different simulators, configurations, or datasets hinders fair benchmarking, as the results are not directly comparable [25].

3. FOG RESOURCE SCHEDULING: PROBLEM FORMULATION

The Fig. 1 typically involves IoT nodes that generate tasks, which are then processed by a set of fog nodes. The evaluation would analyze various performance metrics, such as energy consumption, latency, and throughput. Energy-efficient algorithms aim to reduce power usage, potentially through methods like DVFS. QoS-aware algorithms, on the other hand, focus on ensuring that tasks are completed within their deadlines and with acceptable latency. Power consumption is time-dependent in such systems, which presents challenges in maintaining independent decision sets across time slots. Therefore, we propose a resource scheduling algorithm to find the optimal matching of tasks and resources.

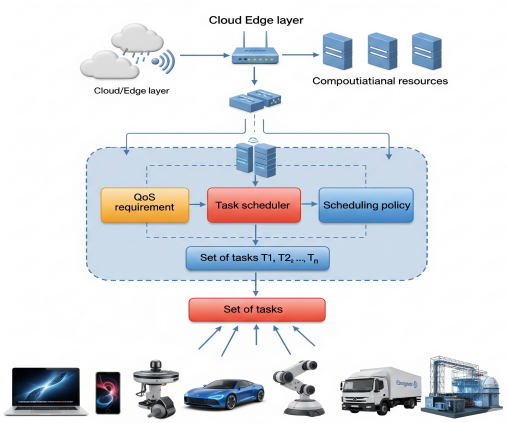

Fig. 1. Fog computing resource-scheduling architecture based on task scheduling

Assume there are N IoT devices that produce a set of tasks, and M available resources. The task set is denoted as $T = t_1, t_2, t_3, \ldots, t_n$, and the set of fog resources is represented as $R = r_1, r_2, r_3, \ldots, r_m$. Each IoT device generates tasks randomly, and it is assumed that the task arrivals are independent and identically distributed. Over a long time period, the task arrival pattern can be modeled using a Poisson distribution. The resource monitor monitors the fog resource pools, such as storage resources, computing resources, and bandwidth resources. If the processing requests of the terminal users to be processed are beyond the computing power of fog computing, these tasks can be submitted by cloud servers for

additional processing when needed. Lastly, on the basis of a particular scheduling strategy, resources and user requests are matched to the corresponding, and the last scheduling outputs will be returned to users.

To address the challenges of efficient task offloading and resource distribution, we propose a novel scheme for Energy-Efficient QoS Fog Evaluation (EEQFE), which leverages a hybrid Whale Optimization Algorithm combined with a Convolutional Neural Network (WOAC). This approach is designed to optimize system performance by minimizing energy consumption while ensuring high Quality of Service (QoS) in fog computing environments.

4. PROPOSED METHODOLOGY

This Proposed methodology WOAC combines the WOA and CNN to optimize fog computing resource scheduling. This is a novel energy-efficient and QoS-aware scheduling algorithm for fog environments. This hybrid approach is designed to balance QoS requirements, such as low latency and high throughput, with energy consumption on fog nodes. The paper highlights the novelty of WOAC by addressing a critical gap in existing research the lack of a comprehensive comparative evaluation framework for assessing diverse scheduling algorithms across both energy and QoS dimensions.

This combines the strengths of the WOA and a CNN to enhance resource scheduling. Unlike traditional approaches the WOAC uses a CNN pre-trained on system state data to predict resource availability or generate intelligent initial task assignments. WOA also utilizes these CNN outputs to optimize a multi-objective function aimed at minimizing latency and energy consumption while maximizing throughput. By leveraging the CNN pattern recognition capabilities for informed initialization and the WOA global search capabilities, WOAC achieves a more refined, adaptive, and efficient solution than single method algorithms.

4.1 Whale Optimization Algorithm with Convolutional Neural Network (WOAC)

Suppose we have a set of fog resources each represented by a dimensional vector e.g., CPU, memory, bandwidth, latency. And there is a set of tasks generated by edge devices:

$$X = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in R^d$$

Each task x_i is defined by features such as CPU requirement, memory, data size, deadline, and priority.

Let there be a set of fog nodes:

$$N = \{n_1, n_2, \ldots, n_m\}$$

The objective is to assign each task x_i to a fog node n_j such that the overall cost function is minimized, subject to resource and QoS constraints. To intelligently represent complex task patterns, we first train a CNN based autoencoder on the task dataset. The encoder maps each task vector x_i into a latent feature vector $F_i \in R^h$, where $h < d$.

Encoder: $E_\theta : R^d \to R^h$ Decoder: $D_\phi : R^h \to R^d$

Loss Function: The reconstruction loss function is defined as:

$$\mathcal{L}_{\text{recon}} = \frac{1}{n} \sum_{i=1}^{n} \|x_i - D_\phi(E_\theta(x_i))\|^2$$

After training, the decoder is discarded and the encoder is used to generate feature vectors:

$$F_i = E_\theta(x_i)$$

These feature vectors are used to initialize the WOA population with more task representation. This represents the bubble-net feeding behavior of humpback whales. Each whale (search agent) represents a possible task-to-node assignment vector. Each whale X_i is a solution:

$$X_i = [a_1, a_2, \ldots, a_n], \quad a_k \in \{1, 2, \ldots, m\}$$

indicating that task x_k is assigned to node n_{a_k}.

WOA updates each solution using three strategies:

a. Encircling Prey (Exploitation):

$$\mathbf{D} = |\check{\mathbf{C}} \cdot \mathbf{X}^* - \mathbf{X}_i|$$
$$\mathbf{X}_i(t+1) = \mathbf{X}^* - \bar{\mathbf{A}} \cdot \mathbf{D}$$

where:

- $\bar{\mathbf{A}} = 2\bar{\mathbf{a}} \cdot \mathbf{7} - \bar{\mathbf{a}}$
- $\check{\mathbf{C}} = 2 \cdot \mathbf{7}, \mathbf{7} \in [0, 1]^d$

b. Spiral Updating (Bubble-net):

$$\mathbf{X}_i(t+1) = \dot{\mathbf{D}} \cdot e^{b\mathbf{l}} \cdot \cos(2\pi\mathbf{l}) + \mathbf{X}^*$$

where:

- $\dot{\mathbf{D}} = |\mathbf{X}^* - \mathbf{X}_i|$, b: spiral coefficient, $\mathbf{l} \in [-1, 1]$

c. Random Search (Exploration):

$$\mathbf{X}_i(t+1) = \mathbf{X}_{\text{rand}} - \bar{\mathbf{A}} \cdot |\check{\mathbf{C}} \cdot \mathbf{X}_{\text{rand}} - \mathbf{X}_i|$$

Executed when $|\bar{\mathbf{A}}| \geq 1$

d. Fitness Function for Evaluation

The fitness of each whale $\mathbf{X}_i$ is determined by a multi-objective function:

$$f(\mathbf{X}_i) = w_1 \cdot \mathbf{L}(\mathbf{X}_i) + w_2 \cdot \mathbf{E}(\mathbf{X}_i)$$

Where:

- $\mathbf{L}(\mathbf{X}_i)$: Total latency of task execution and communication.
- $\mathbf{E}(\mathbf{X}_i)$: Total energy consumed by all fog nodes.
- w_1, w_2, w_3: Weights based on system priorities.

The optimization objective is to find the optimal solution $\mathbf{X}^*$ that minimizes the fitness function $f(\mathbf{X}_i)$:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}_i} f(\mathbf{X}_i)$$

$X \in [1, 2, 3, \ldots, n]$ position of i-th whale

X^* is the optimal solution

$f(\mathbf{X}_i)$ Fitness value of solution $\mathbf{X}_i$

4.2 Energy Efficient Model

However, the problem states that we assign tasks to fog nodes. Therefore, we can model the energy for a fog node as the energy consumed in processing the tasks assigned to it and communicating with the edge devices. Since the fog node receives data from edge devices, the energy for receiving the data of task x_k

$$E_{rx}(k,j) = \text{data}_k \cdot P_{rx}^j$$

The time to compute task x_k on node n_j is

$$t_{\text{comp}}^k = \frac{\text{cpu}_k}{f_j}$$

The energy computation during the time is

$$E_{\text{comp}}(k,j) = P_{\text{active}}^j \cdot t_{\text{comp}}^k$$

Assuming the result size is proportional to the input data is

$$\text{result}_k = \beta \cdot \text{data}_k$$

where β is some factor. Then the transmission energy is

$$E_{\text{tx}}(k,j) = \text{result}_k \cdot P_{\text{tx}}^j$$

The energy for task k on fog node j is:

$$E_{k,j} = \text{data}_k \cdot P_{\text{rx}}^j + P_{\text{active}}^j \cdot \frac{\text{cpu}_k}{f_j} + \beta \cdot \text{data}_k \cdot P_{\text{tx}}^j$$

Then, the total energy for node j is the sum over all tasks assigned to it and the total energy for the fog node n_j is:

$$E_j = \sum_{k \in \mathcal{T}} \left[\text{data}_k \cdot (P_{\text{rx}}^j + \beta \cdot P_{\text{tx}}^j) + P_{\text{active}}^j \cdot \frac{\text{cpu}_k}{f_j} \right]$$

Then, the total energy for the system (all fog nodes) is:

$$E(X_i) = \sum_{j=1}^{m} E_j = \sum_{j=1}^{m} \sum_{k \in \mathcal{T}} \left[data_k \cdot (P_j^{Tx} + \beta \cdot P_j^{Tx}) + P_j^{active} \cdot \frac{cpu_k}{f_j} \right]$$

Alternatively, we can write it as a sum over tasks. Let a_k be the node index to which task k is assigned. Then:

$$E(X_i) = \sum_{k=1}^{n} \left[data_k \cdot (P_k^{Tx} + \beta \cdot P_k^{Tx}) + P_k^{active} + \frac{cpu_k}{f_k} \right]$$

We introduced a parameter β for the result data size. If we do not want to consider the result transmission, we can set $\beta = 0$. Alternatively, if the result size is known, we can adjust β accordingly. Also, note that the parameters $P_j^{rx}, P_j^{tx}, P_j^{\text{active}}, f_j$ are properties of the fog node n_j and are known. Therefore, the energy model is:

$$E(X_i) = \sum_{k=1}^{n} \left[data_k \cdot (P_{a_k}^{Tx} + \beta \cdot P_{a_k}^{Tx}) + P_{a_k}^{active} + \frac{cpu_k}{f_{a_k}} \right]$$

4.3 Latency Model

In the context of task offloading to fog computing environments, minimizing latency is a critical performance objective. We define the end-to-end latency for a given task as the sum of three distinct components: Transmission Latency (TL), Processing Latency (PL), and Propagation Latency (PRL). Each component accounts for different delays encountered by a task from its initiation to its completion at a fog node.

4.3.1 Transmission Latency (TL): Transmission Latency (TL) refers to the time required to transfer the data associated with task k from the edge device to its assigned fog node j. This delay is directly proportional to the size of the task's data and the number of tasks concurrently utilizing the bandwidth of the fog node. We model TL_k as follows:

$$TL_k = \frac{\text{data\_size}_k \cdot |T_j|}{\text{BW}_j}$$

where data_size_k: Represents the data size of task k. $|T_j|$ denotes the number of tasks currently assigned to fog node j. This term captures the shared nature of the fog node's bandwidth, implying that as more tasks are assigned to a node, the effective bandwidth available per task decreases, thus increasing transmission latency. Represents the total available bandwidth of fog node j.

4.3.2 Processing Latency (PL): Processing Latency (PL) quantifies the time taken for fog node j to execute task k. This latency is contingent upon the computational demands of the task and the processing capacity of the assigned fog node. Similar to transmission latency, the processing resources are shared among all tasks assigned to a specific node. The formula for PL_k is given by:

$$PL_k = \frac{\text{cpu}_k \cdot |T_j|}{\text{cpu}_j}$$

where cpu_k refers to the CPU requirement of task k. $|T_j|$: As before, represents the number of tasks assigned to fog node j. This factor highlights the impact of load on the processing capabilities, where increased load leads to longer processing times per task. cpu_j: represents the total CPU capacity of fog node j.

4.3.3 Propagation Latency (PRL): Propagation Latency ($Prop_k$) accounts for the fixed network delay experienced by data signals traveling between the edge device and the fog node. This component typically depends on the physical distance and the transmission medium characteristics. For simplicity, we assume this is an inherent feature of the fog node's network connectivity. It is defined as:

$$Prop_k = \text{latency}_j$$

where latency_j Represents the inherent fixed propagation delay associated with fog node j. This value can be predetermined based on the geographical location and network infrastructure of the fog node.

Therefore the total latency $Latency_k$ of k tasks is the sum of its transmission, processing, and propagation latencies:

$$\text{Latency}_k = TL_k + PL_k + PRL_k$$

Therefore the fitness function is to optimize the problems of minimize overall system latency, a fitness function is employed to evaluate the quality of a given task assignment strategy. Our objective is to minimize the sum of latencies for all tasks k within the system. For a given assignment X_i (representing a specific allocation of n tasks to available fog nodes), the total latency, $L(X_i)$, is defined as:

$$L(X_i) = \sum_{k=1}^{n} \left(\frac{\text{data\_size}_k \cdot |T_j|}{\text{BW}_j} + \frac{\text{cpu}_k \cdot |T_j|}{\text{cpu}_j} + \text{latency}_j \right)$$

The inclusion of the term $|T_j|$ in both the transmission and processing latency components serves as a dynamic penalty for overloaded nodes. This crucial aspect of the fitness function promotes effective load balancing across the available fog nodes. By lower the latency contribution of tasks assigned to heavily utilized nodes, the optimization algorithm is guided towards distributing tasks more evenly, thereby preventing bottlenecks and improving overall system performance.

5. Algorithm Design

Algorithm 1 Resource Scheduling Algorithm Based on WOAC

Require: Resource set $\{rs_1, rs_2, \ldots, rs_m\}$, task set $\{ta_1, ta_2, \ldots, ta_n\}$.
max itr T_{max}, WOA size N, CNN prediction probability P_{cnn}, pre-trained CNN model

Ensure: Best energy consumption E_{best}, latency L_{best}, and QoS score Q_{best}

Initialize whale population $\{X_i\}_{i=1}^{N}$
Set $E_{\text{best}} \leftarrow \infty$, $L_{\text{best}} \leftarrow \infty$, $Q_{\text{best}} \leftarrow 0$, $t \leftarrow 0$
while $t < T_{\text{max}}$ **do**
  $a \leftarrow 2\left(1 - \frac{t}{T_{\text{max}}}\right)$
  for each whale X_i **do**
    Update X_i using WOA position update rules
    if rand $< P_{\text{cnn}}$ **then**
      $E_i \leftarrow \text{CNN\_Predict\_Energy}(X_i)$
      $L_i \leftarrow \text{CNN\_Predict\_Latency}(X_i)$
      $Q_i \leftarrow \text{CNN\_Predict\_QoS}(X_i)$
    else
      $(E_i, L_i, Q_i) \leftarrow \text{Simulate}(X_i)$
    end if
    if $E_i < E_{\text{best}}$ **and** $L_i < L_{\text{best}}$ **and** $Q_i > Q_{\text{best}}$ **then**
      $E_{\text{best}} \leftarrow E_i$, $L_{\text{best}} \leftarrow L_i$, $Q_{\text{best}} \leftarrow Q_i$
      $X_{\text{best}} \leftarrow X_i$
    end if
  end for
  $t \leftarrow t + 1$
end while
return $E_{\text{best}}, L_{\text{best}}, Q_{\text{best}}$

5.1 Algorithm Performance Analysis

The efficacy of our Proposed WOAC for resource scheduling is shaped by several core elements. The computational cost scales with the total iterations (T_{max}) and the population count (N), as each step requires either a swift CNN inference or a potentially costly simulation. Convergence speed benefits from the WOA inherent search capabilities, amplified by the CNN predictive guidance. However, the achieved solution improvement in energy, latency, and QoS hinges critically on both the WOA exploration and the CNN precision; a high P_{cnn} with an imprecise CNN model can degrade outcomes. Thus, the pre-trained CNN quality is paramount, as an inadequately trained model will misdirect the optimization. The Simulate(X_i) function, when invoked, contributes notable computational overhead. Furthermore, performance is sensitive to tuning parameters like T_{max}, N, and P_{cnn}. Scalability depends on the CNN and simulation complexity relative to problem size. While simulation validation can bolster robustness, an over-reliance on a non-robust CNN especially with high P_{cnn} exposes the algorithm to vulnerabilities in novel situations.

TABLE II
COMPARISON OF ALGORITHMS BASED ON NUMBER OF TASKS, TS ENERGY IN(J), AND TT ENERGY(J))

Algorithm	Number of Tasks	TS Energy(J)	TT Energy(J)
WOAC	100	210.57	139.97
	200	353.85	265.90
	300	581.82	426.80
	400	805.09	601.19
	500	939.62	749.56
ABC	100	247.90	171.96
	200	469.89	360.05
	300	674.08	555.44
	400	939.06	728.60
	500	1183.23	942.92
FCFS	100	283.20	203.94
	200	540.12	400.00
	300	770.10	611.11
	400	1091.11	776.34
	500	1287.23	989.21

The table II presents the performance comparison between two distinct optimization algorithms, WOAC, FCFS and ABC specifically in the context of energy consumption metrics for varying computational loads. The number of tasks serves as a proxy for the computational workload, ranging from 100 to 500 tasks where the fix same fog nodes 50. This allows for an analysis of how each algorithm scales under increasing demands. The primary performance indicators are Total System (TS) energy and Total Tasks (TT) energy consumption in Jules both reflecting different aspects of energy expenditure during task execution.

The Fig 2. represents the total energy consumption of tasks in (J/sec). It compares the WOAC, FCFS, and ABC algorithms' energy use as task count increases (100-500). While both consume more energy with more tasks, WOA consistently shows lower total consumption than ABC across all loads, demonstrating greater energy efficiency.

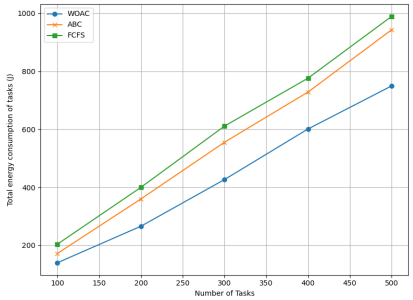

Fig. 2. Total energy consumption of tasks (TEC).

Fig 3. shows the TS energy across three algorithms is that as the number of tasks grows, both types of energy consumption rise proportionally, indicating that higher workloads demand more energy. Furthermore, it shows the comparison of results are in ABC, FCFS and our proposed approach. It shows the values of ABC consistently exhibits higher than WOAC. Therefore, WOAC demonstrates better energy efficiency than ABC under the evaluated conditions.

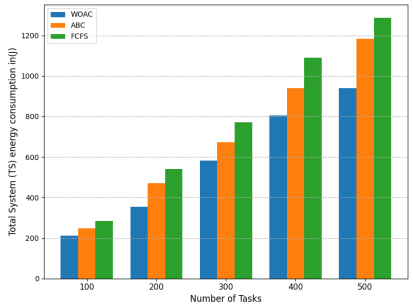

Fig. 3. Comparision of TS energy in all approaches.

TABLE III
COMPARISON OF ALGORITHMS BASED ON NUMBER OF TASKS TOTAL LATENCY, PL, AND TL

Algorithm	Tasks	Total Latency (TL)	PL	TL
WOA	100	65.5	27.9	37.3
	150	96.2	40.1	56.1
	200	132.4	53.1	69.2
	250	174.6	76.7	90.8
	300	192.3	85.3	107.0
ABC	100	72.4	34.3	38.8
	150	109.6	52.3	57.5
	200	146.6	70.25	74.6
	250	190.8	94.3	96.5
	300	222.5	111.0	114.4
FCFS	100	79.25	35.5	43.5
	150	113.92	53.5	59.5
	200	154.10	72.4	80.23
	250	200.45	95.21	105.69
	300	232.40	110.6	121.4

The above Table III shows the Comparison of WOAC, ABC and FCFS algorithms based on latencies. It details TL, PL and TL for task numbers ranging from 100 to 300. For each task load, all algorithms latency values are provided across these three categories. The data consistently indicates that WOAC achieves lower latency values in all measured categories compared to ABC and FCFS.

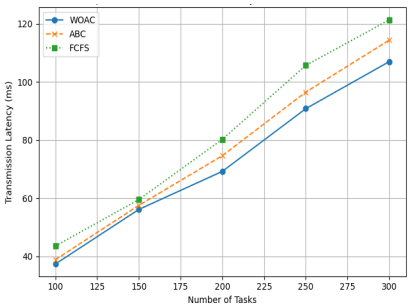

Fig. 4. Comparison of TL in both approaches

In Fig 4. we define the correlation between the volume of tasks and TL for the WOAC, ABC and FCFS algorithms. When the number of task count rises, the TL increases for both approaches. Nevertheless, the WOAC algorithm consistently demonstrates lower latency across the entire spectrum of task loads. This indicates that WOAC is more effective for data transmission in computational environments handling a substantial number of tasks.

In Fig 5 we define clearly the WOAC algorithm is superior in processing efficiency as compared to ABC and FCFS. Despite both algorithms experiencing increased latency with more tasks, WOAC consistently maintains significantly lower processing delays. This crucial benefit makes WOAC highly effective for rapid task execution, particularly in scenarios involving a large number of tasks.

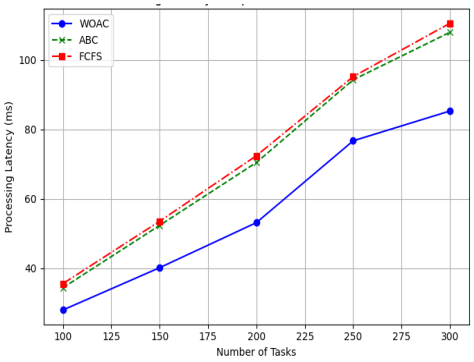

Fig. 5. Comparison of PL in both approaches

Fig 6 We shows the graph total latency performance of the WOAC and ABC across varying task loads. Both algorithms exhibit increased latency with higher task counts; however, WOAC consistently achieves significantly lower overall delays. This result empirically demonstrates WOAC superior efficiency and enhanced scalability in managing substantial workloads. The reduced latency provided by WOAC is particularly critical for time-sensitive applications, such as real-time systems, enabling improved system responsiveness, enhanced user experience, and more effective resource utilization.

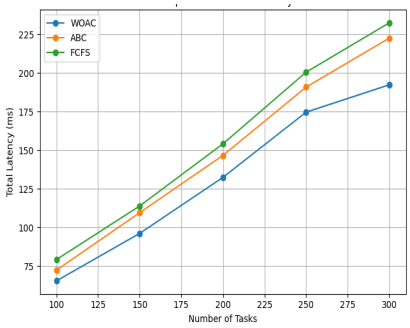

Fig. 6. Comparison of Total Latency in both approaches

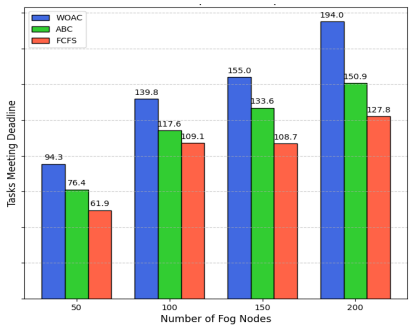

Fig. 8. Comparison of tasks meeting deadline

TABLE IV
COMPARISON OF WOAC AND ABC ALGORITHMS BASED ON THROUGHPUT AND DEADLINE MEETING TASKS

Algorithm	Fog Nodes	Throughput	Task Deadline
WOAC	50	95.9	94.3
	100	141.9	139.8
	150	155.5	155.0
	200	194.0	194.0
ABC	50	78.78	76.39
	100	118.80	117.61
	150	135.59	133.56
	200	150.88	150.88
FCFS	50	61.77	61.9
	100	110.85	109.1
	150	115.4	108.67
	200	129.51	127.78

The Fig 7. shows a comparison of throughput evaluation contrasting the WOAC and ABC algorithms. The fog nodes are increasing The WOAC algorithm consistently outperforms the ABC algorithm, with a maximum throughput of 194.0 at 200 fog nodes compared to 150.88 for the ABC are shown in Table IV. This comparison underscores WOAC enhanced throughput capabilities, positioning it as advantageous for edge-based applications demanding high-performance data processing.

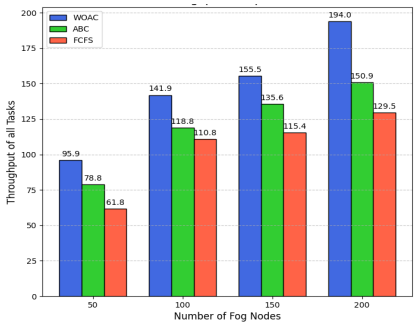

Fig. 7. Comparison of Throughput

Fig 8. compares deadline compliance between WOAC and ABC, FCFS algorithms as fog nodes scale. WOAC consistently achieves higher compliance, peaking at 194.0 tasks meeting deadlines at 200 node as compared ABC and FCFS only 150.88, and 127.78. This clears the WOAC is effective for real-time applications where meeting deadlines is critical.

6. CONCLUSIONS

This paper presents a comprehensive comparative performance evaluation of leading energy-efficient and QoS-aware scheduling algorithms in fog computing environments. The results reveal that the proposed hybrid WOAC algorithm consistently outperforms the ABC and FCFS algorithm across key metrics, including energy efficiency, latency reduction, throughput, and deadline adherence. Specifically, WOAC achieves superior energy savings and enables faster data processing and more efficient communication, making it particularly well-suited for time-sensitive applications. Additionally, WOAC delivers significantly higher task processing rates, with the performance advantage becoming more pronounced as the number of fog nodes increases. However, it maintains an almost perfect correlation between overall throughput and on-time task completion. These findings highlight WOAC as a highly scalable and dependable solution for fog computing scenarios that demand both high throughput and strict deadline compliance.

REFERENCES

[1] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments: A taxonomy, review and future directions," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–43, 2020.

[2] G. Li, Y. Liu, J. Wu, D. Lin, and S. Zhao, "Methods of resource scheduling based on optimized fuzzy clustering in fog computing," *Sensors*, vol. 19, no. 9, p. 2122, 2019.

[3] G. Goel and R. Tiwari, "Resource scheduling techniques for optimal quality of service in fog computing environment: a review," *Wireless Personal Communications*, vol. 131, no. 1, pp. 141–164, 2023.

[4] A. Rahimikhanghah, M. Tajkey, B. Rezazadeh, and A. M. Rahmani, "Resource scheduling methods in cloud and fog computing environments: a systematic literature review," *Cluster Computing*, vol. 25, no. 2, pp. 911–945, 2022.

[5] C. Huang, H. Wang, L. Zeng, and T. Li, "Resource scheduling and energy consumption optimization based on lyapunov optimization in fog computing," *Sensors*, vol. 22, no. 9, p. 3527, 2022.

[6] F. Murtaza, A. Akhunzada, S. ul Islam, J. Boudjadar, and R. Buyya, "Qos-aware service provisioning in fog computing," *Journal of Network and Computer Applications*, vol. 165, p. 102674, 2020.

[7] N. Potu, C. Jatoth, and P. Parvataneni, "Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 23, p. e6163, 2021.

[8] M. Iyapparaja, N. K. Alshammari, M. S. Kumar, S. Krishnan, and C. L. Chowdhary, "Efficient resource allocation in fog computing using qtcs model." *Computers, Materials & Continua*, vol. 70, no. 2, 2022.

[9] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–38, 2022.

[10] C. Yin, Q. Fang, H. Li, Y. Peng, X. Xu, and D. Tang, "An optimized resource scheduling algorithm based on ga and aco algorithm in fog computing," *The Journal of Supercomputing*, vol. 80, no. 3, pp. 4248–4285, 2024.

[11] X. Li, Z. Zang, F. Shen, and Y. Sun, "Task offloading scheme based on improved contract net protocol and beetle antennae search algorithm in fog computing networks," *Mobile Networks and Applications*, vol. 25, no. 6, pp. 2517–2526, 2020.

[12] M. A. Rahman, M. S. Hossain, E. Hassanain, and G. Muhammad, "Semantic multimedia fog computing and iot environment: sustainability perspective," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 80–87, 2018.

[13] Y. Kalyani and R. Collier, "A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture," *Sensors*, vol. 21, no. 17, p. 5922, 2021.

[14] X. Huang, W. Fan, Q. Chen, and J. Zhang, "Energy-efficient resource allocation in fog computing networks with the candidate mechanism," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8502–8512, 2020.

[15] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Information Systems*, vol. 12, no. 4, pp. 373–397, 2018.

[16] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE internet of things journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[17] M. Mukherjee, S. Kumar, C. X. Mavromoustakis, G. Mastorakis, R. Matam, V. Kumar, and Q. Zhang, "Latency-driven parallel task data offloading in fog computing networks for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6050–6058, 2019.

[18] H. Sabireen and V. Neelanarayanan, "A review on fog computing: Architecture, fog with iot, algorithms and research challenges," *Ict Express*, vol. 7, no. 2, pp. 162–176, 2021.

[19] Y. Liu, X. Zeng, Z. He, and Q. Zou, "Inferring microrna-disease associations by random walk on a heterogeneous network with multiple data sources," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 14, no. 4, pp. 905–915, 2016.

[20] M. Haghi Kashani, A. M. Rahmani, and N. Jafari Navimipour, "Quality of service-aware approaches in fog computing," *International Journal of Communication Systems*, vol. 33, no. 8, p. e4340, 2020.

[21] G. Caiza, M. Saeteros, W. Oñate, and M. V. Garcia, "Fog computing at industrial level, architecture, latency, energy, and security: A review," *Heliyon*, vol. 6, no. 4, 2020.

[22] Z. Lin, L. Lu, J. Shuai, H. Zhao, and A. Shahidinejad, "An efficient and autonomous planning scheme for deploying iot services in fog computing: A metaheuristic-based approach," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 1, pp. 1415–1429, 2023.

[23] C. Tameling, S. Stoldt, T. Stephan, J. Naas, S. Jakobs, and A. Munk, "Colocalization for super-resolution microscopy via optimal transport," *Nature computational science*, vol. 1, no. 3, pp. 199–211, 2021.

[24] A. Brogi, S. Forti, and A. Ibrahim, "Optimising qos-assurance, resource usage and cost of fog application deployments," in *International conference on cloud computing and services science*. Springer, 2018, pp. 168–189.

[25] S. Sonkamble, S. Chandra, and P. R. Pujari, "Application of airborne and ground geophysics to unravel the hydrogeological complexity of the deccan basalts in central india," *Hydrogeology Journal*, vol. 30, no. 7, pp. 2097–2116, 2022.