AI-based Sustainable and Intelligent Offloading Framework for IIoT in Collaborative Cloud-Fog Environments

Mohit Kumar, Guneet Kaur Walia, Haresh Shingare, Samayveer Singh, Sukhpal Singh Gill

***Abstract*— The cloud paradigm is one of the most trending areas in today's era due to its rich profusion of services. However, it fails to serve the latency-sensitive Industrial Internet of Things (IIoT) applications associated with automotives, robotics, oil and gas, smart communications, Industry 5.0, etc. Hence, to strengthen the capabilities of IIoT, fog computing has emerged as a promising solution for latency-aware IIoT tasks. However, the resource-constrained nature of fog nodes puts forth another substantial issue of offloading decisions in resource management. Therefore, we propose an Artificial Intelligence (AI)-enabled intelligent and sustainable framework for an optimized multi-layered integrated cloud fog-based environment where real-time offloading decisions are accomplished as per the demand of IIoT applications and analyzed by a fuzzy based offloading controller. Moreover, an AI based Whale Optimization Algorithm (WOA) has been incorporated into a framework that promises to search for the best possible resources and make accurate decisions to ameliorate various Quality-of-Service (QoS) parameters. The experimental results show an escalation in makespan time up to 37.17%, energy consumption up to 27.32%, and execution cost up to 13.36% in comparison to benchmark offloading and allocation schemes.**

***Index Terms*—Task offloading, Internet of Things (IoT), Fog computing, Resource Allocation, Artificial Intelligence (AI)**

I. INTRODUCTION

With the digital infrastructure revolutionizing the world at an expeditious rate, the Industrial Internet of Things (IIoT) is emerging rapidly, embracing applications such as automotives, smart cities, healthcare, waste and water management, robotics, smart communication, smart power grids etc. These Industry 5.0 based use cases thrive on resources to process, analyze and store this colossal amount of data generated by IIoT applications. In contrast to IoT applications such as handheld devices which are consumer-centric in nature, IIoT applications emphasize more on achieving sustainability, ensuring health and safety, and ameliorating overall system efficiency.

However, the unprecedented advancement in terms of high-speed networking capabilities such as 5G and beyond, pervasive computing devices, mobile applications, and IoT sensors are generating tremendous amounts of data that require real-time analytics. For instance, consider the Industry 5.0 scenario, which is emerging with the concept of Collaborative Robots (COBOTS) which learn to work with humans in a collaborative manner. In contrast to industrial robots which are designated to perform specific tasks, COBOTS are equipped with intelligence to perform a diverse range of tasks with humans in a cooperative manner, ensuring safety and ameliorating the productivity of enterprises. Despite its eminent capabilities, any robotic malfunction due to delayed response could make the situation worse. Hence, sending the data perceived from IIoT sensors to centralized cloud datacenters is not a practical solution.

Henceforth, a novel distributed paradigm known as Fog Computing (FC) has emerged, which leverages cloud characteristics with new features like location awareness and edge datacenter deployment. It refers to computing at the edge of the network, enabling distributed computing solutions in order to maximize scalability, elasticity, resiliency, minimizing computational costs, and efficient information sharing, among other things. [1]. However, its complicated decentralized architecture imposes various challenges in order to effectively utilize the underlying heterogeneous resources.

The resource management issues in the fog landscape comprise resource provisioning, task offloading, task mapping, and service placement. In recent works, some of the researchers have proposed solutions for optimal resource surveillance and management mechanisms. However, ensuring optimality of task offloading decisions is quite crucial in a collaborative Fog-Cloud environment for Industry 5.0 use cases. The complexity of this decision comprises many factors such as which task to offload, finalizing the offloading destination (Fog Nodes or a cloud datacenter instance) and, when to offload the task [2]. The delay intensive tasks are offloaded to Fog Nodes (FNs), consequently the processing and transmission delays arereduced to meet the Quality of Service (QoS) constraints. To serve these capabilities, our work aims to propose an effective task offloading strategy

Mohit Kumar is with the Department of Information Technology, NIT Jalandhar, Punjab 144027, India. E-mail: kumarmohit@nitj.ac.in
Guneet Kaur Walia is with the Department of Information Technology, NIT Jalandhar, Punjab 144027, India. E-mail: guneetkw.it.22@nitj.ac.in
Haresh Shingare is with the Department of Computer Science and Engineering, NIT Jalandhar, Punjab 144027, India. E-mail: hareshss.cs.20@nitj.ac.in
Samayveer Singh is with the Department of Computer Science and Engineering, NIT Jalandhar, Punjab 144027, India. E-mail: samays@nitj.ac.in
Sukhpal Singh Gill is with School of Electronic Engineering and Computer Science, Queen Mary University of London, U.K, E-mail: s.s.gill@qmul.ac.uk

which satisfy the QoS parameters such as optimizing makespan, cost and task rejection ratio.

A. MOTIVATION

Offloading incoming tasks over an optimal paradigm becomes a significant aspect of resource management issues in an IIoT-enabled collaborative cloud-fog environment which directly impacts sustainability, resilience and other QoS parameters. The offloading decision is based upon several factors like incoming task demand, size, deadline and availability of fog resources. When, why and where to offload the upcoming IIoT task is a challenging issue due to the unpredictability, fluctuation, and diverse nature of the IoT workload. Communication cost and mobility of nodes in a dynamic environment make it a harder problem that does not provide the guarantee of optimized QoS parameters [3]. Hence, there is a dire need for an intelligent and dynamic algorithm to address the mentioned issues and evaluate the cost after offloading decisions (locally or remotely execution) [4]. In addition, the issue of sustainability remains a challenging task in Industry 5.0 due to ever-increasing energy costs and their never-ending escalating demands, which thrive on novel techniques for smart energy systems [5]. To ensure that the challenges are met, our work envisions proposing a fuzzy-based task offloading decision and a meta-heuristic based Whale Optimization Algorithm (WOA) has been applied for IIoT workload allocation. The proposed technique overcomes the drawbacks of traditional metaheuristic algorithms: getting trapped in local optimality, slow convergence, and poor exploration and exploitation. Moreover, our suggested task offloading scheme addresses the issue of sustainability along with optimizing significant QoS parameters.

B. OUR CONTRIBUTIONS

The primary contributions of this work are:

- To formulate a task offloading problem for Industry 5.0 deadline-aware workloads that need to be serviced in a collaborative Cloud-Fog computing environment. The problem takes into account the dynamics of the incoming request and aims to minimize the long-term cost associated with the optimal task offloading decision.
- To achieve expected cost minimization, an AI-enabled decision-making offloading framework based on fuzzy models and metaheuristics is proposed. The former intelligently decides whether to execute tasks locally (Edge device) or remotely (on a FN or cloud datacenter). A metaheuristic approach known as the Whale Optimization Algorithm (WOA) is incorporated for optimal task-to-resource mapping in a collaborative cloud fog scenario.
- The extensive experimental analysis justifies that the proposed approach shows significant improvements procured in terms of makespan, execution cost, rejection ratio, and energy consumption in comparison with benchmark techniques.

The rest of the paper is organized as follows: Section II discusses the state-of-the-art work done in the arena of offloading and resource allocation. Section III depicts the problem formulation of the proposed framework. The detailed architectural framework along with the proposed offloading and resource allocation algorithms of our study have been illustrated in Section IV and V, respectively. Section VI discusses the simulation results considering distinctive performance attributes along with the experimental setup. Section VII represents a summary of the proposed work along with future work.

II. RELATED WORK

This section illustrates the work done in the arena of resource scheduling and monitoring utilizing AI and non-AI based techniques. In the work done by Nan et al. [6], an algorithm known as Lyapunov Optimization on Time and Energy Cost (LOTEC) is proposed to investigate energy-efficient data processing to balance the time it takes to analyze the information vs. the cost incurred in running the system. Based on the simulation results, the proposed strategy looks promising.

The fog paradigm is envisioned as a practicable solution for extending the resource-constrained domain of IIoT devices and concurrently, improving dynamic workload performance. This paper demonstrates task prioritization using fuzzy logic based on resource requirements and the deadline to be met. The suggested technique reduces waiting time by 35% and service latency by 28%, respectively [7]. Nevertheless, task deadlines play a vital role in making optimal task offloading decisions. In this context, Adhikari et al. [8], discussed the importance of completing hard-deadline tasks on time as having mission criticality. Hence, such tasks need to be processed in close proximity to the service requestor node, for which a rule-based task scheduling strategy is proposed to frame an apt task sequence while minimizing waiting time.

The latest research in fog computing utilizes nature-inspired metaheuristic algorithms. These are computationally intelligent algorithms that are effectively capable of solving sophisticated optimization problems. Goyal S. et al. [9] used PSO, CSO, BAT, CSA and WOA to improve multiple QoS parameters. This study emphasized load balancing which implements uniform distribution of load over multiple servers to satisfy consumers' growing demands. In another study [10] authors proposed the usage of the Crow Search Algorithm (CSA) with the Deep Auto Encoder (DAE) in order to perform

autonomic workload prediction and allocate resources to incoming IIoT requests. The authors in [11] emphasized the importance of task offloading in smart vehicular systems, comprise of multiple Roadside Units (RSUs) near fog servers and are placed to serve a variety of incoming traffic such as infotainment, emergency, and traffic management using a knapsack-based task scheduling algorithm. Abdel-Basset et al. [12] proposed the Modified Marine Predators Algorithm (MMPA) for tackling Task Scheduling in Fog Computing (TSFC) amongst IoT applications. Another work implemented DDN task partitioning and offloading in Vehicular Edge Computing (VEC), however the work did not emphasize the sustainability aspect in the considered scenario [13]. Apart from VEC, a fog-assisted framework has been proposed for data aggregation at FNs in order to ensure prompt delivery in the Internet of Medical Things(IoMT) [14].

As depicted from the literature surveyed, it has been observed that majority of studies failed to achieve an efficient offloading decision and were unable to propose a plausible solution for optimal resource allocation to the upcoming IIoT use cases. In addition, the majority of the works have proposed offloading approaches either for cloud or fog, an integrated environment has not been considered for real-time use cases.

III. PRELIMINARIES

This section equips the researchers with the list of notations used in problem formulation and computational models as depicted in Table 2.

Table 2: Preliminaries for Work

Notation	Description
$\mathfrak{D}$, $\mathcal{F}$	Set of IoT devices and FNs
ß$\mathbf{W}_{\kappa}^{\mathbf{IF}}$	Bandwidth between IoT device and Fog Node
ß$\mathbf{W}_{\kappa}^{\mathbf{CI}}$	Bandwidth between cloud and IoT device
$\mathcal{T}_{\kappa}$	Task size of incoming smooth and bursty workload
$\mathfrak{R}_{\mathcal{M}}$	Total resources of comprising $\mathcal{M}$ computational units
$\rho_k^{ex\rightarrow IoT}$	Power for executing task at local device
$\rho_k^{ex\rightarrow \mathcal{CD}}$	Power for executing task at cloud datacenter
$\rho_k^{ex\rightarrow \mathcal{FN}}$	Power for executing task at FN
TD_{LAN}	Transmission Delay between IoT and FN
TD_{WAN}	Transmission Delay between IoT and cloud
$\mathcal{E}\Gamma_{\mathcal{T}_k}^{IoT}$	Time spent to execute $\mathcal{T}_{\kappa}$ at Fog node
$\mathcal{E}\Gamma_{\mathcal{T}_k}^{\mathcal{CD}}$	Time spent to execute $\mathcal{T}_{\kappa}$ at Cloud datacenter
$\mathcal{E}\Gamma_{\mathcal{T}_k}^{\mathcal{FN}}$	Time spent to execute $\mathcal{T}_{\kappa}$ at FN
ξ_k^{IoT}	EC in processing κ_{th}task at IoT device
$\xi_k^{\mathcal{FN}}$	EC in processing task at $\mathcal{FN}$
$\xi_k^{\mathcal{CD}}$	EC in processing κ_{th} task at $\mathcal{CD}$
$\mathbb{C}_{d_{Total}}$(t)	Total processing cost
$\xi_{d_{Total}}(t)$	Total Energy Consumption

A. MODEL FOR OFFLOADING DECISION

The fog layer bridges the gap between the IoT and cloud layers by reducing the round-trip time (RTT) in order to satisfy the user experience of real time IIoT applications. Here, the time is discretized into *T* time slots, where *t* depicts a specific time slot. The set of computational requests generated by IoT devices can be depicted into a series of input tasks $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots \mathcal{T}_n\}$ where $\mathcal{T}_{\kappa(t)}$ represents size of $\mathcal{K}^{\text{th}}$ task in bits. The task sources are symbolized as $\mathfrak{D}$ constituting set of n devices $\mathfrak{D} = \{\mathfrak{D}_1, \mathfrak{D}_2, \mathfrak{D}_3, \ldots, \mathfrak{D}_n\}$ and FNs are expressed as $\boldsymbol{\mathcal{F}} = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \ldots, \mathcal{F}_m\}$ which are geographically distributed around the fog layer. In addition, the sum total of resources available is denoted by $\mathcal{R}_{\mathcal{M}}$, where $\mathcal{M}$ represents number of computational units. Hence, the work aims to predict optimal task offloading decision which allocates set of incoming tasks $\mathcal{T}_n$ to the IIoT devices, FNs and cloud servers. The decision ensures that IIoT devices consume a minimal amount of energy and that QoS parameters constituting makespan time, task rejection, and cost are minimized. An integer task offloading indictor φ_k is used to represent offloading decision corresponding to κ_{th} task generated by IoT device, which is mathematically defined as follows:

$$\varphi_k(t) = \begin{cases} 0\,; when\ \mathcal{T}_k\ is\ execute\ at\ IoT\ device \\ 1\,;\ when\ \mathcal{T}_k\ is\ offloaded\ to\ \mathcal{F}_j \\ -1; when\ \mathcal{T}_k\ is\ offloaded\ to\ \mathcal{CD} \end{cases} \quad (1)$$

Where, $\varphi_k(t) = 0$ signifies task $\mathcal{T}_k$ is processed at IoT device itself. The case of $\varphi_k(t) = 1$ means request is offloaded to local server for latency-sensitive task whereas $\varphi_k(t) = -1$ represents request being offloaded to central server for compute and resource-intensive task.

B. COMPUTATIONAL MODEL

Local Computation Model: Sensors and actuators embedded in various things including cameras, wearables etc. have computational bottlenecks due to their resource-constrained nature. Moreover, these devices are battery driven, hence acquiring limited processing capabilities and limited storage [15]. The execution time of task $\mathcal{T}_{\kappa}$ measures the time spent by IIoT device for processing the incoming task locally, is represented as

$$\mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{IoT}(t) = \frac{\mathcal{T}_{\kappa}(t)}{\omega_{IoT}}\,;\ \forall \mathcal{T}_{\kappa} \in \mathcal{T} \quad (2)$$

where $\mathcal{T}_{\kappa}$ represents the incoming task size of incoming and ω_{IoT} depicts the CPU frequency of local processing unit. Correspondingly, the energy consumption constitutes the energy engrossed by IIoT device while processing the task $\mathcal{T}_{\kappa}$ is expressed as:

$$\xi_{\kappa}^{IoT}(t) = \rho_{\kappa}^{ex\rightarrow IoT}(t) * \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{IoT}(t) \quad (3)$$

Where $\rho_{\kappa}^{ex \to IoT}$ denotes power consumed while executing task at local device. The total processing cost is calculated as,

$$\mathbb{C}_{\kappa}^{IoT}(t) = \theta_1 * \ \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{IoT}(t) + \theta_2 * \xi_{\kappa}^{IoT}(t) \quad (4)$$

Where θ_1 and θ_2 are weight coefficients, in such a way that $\theta_1 + \theta_2 = 1$.

Fog Computational Model: In case the local IIoT device is incompetent to handle the incoming workload, the tasks are offloaded to the fog layer. The execution time of offloading task $\mathcal{T}_{\kappa}$ to FN can be calculated as:

$$\mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{FN}(t) = \frac{\mathcal{T}_{\kappa}(t)}{\omega_{FN}} + TD_{LAN}(t) \ ; \forall \mathcal{T}_{\kappa} \in \mathcal{T} \quad (5)$$

Here, $\mathcal{T}_{\kappa}$ signifies the task length and ω_{FN} represents the computing frequency of FN. Another delay known as transmission delay, TD_{LAN} is added to denote delay by transmission medium, which is as follows:

$$TD_{LAN}(t) = \frac{\mathcal{T}_{\kappa}(t)}{ßW_{\kappa}^{IF}} \quad (6)$$

In addition, the Energy Consumed (EC) during offloading task from IIoT to FN is computed as follows:

$$\xi_{\kappa}^{FN}(t) = \rho_{\kappa}^{ex \to FN}(t) * \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{FN}(t) \quad (7)$$

The total processing cost at this layer is calculated as,

$$\mathbb{C}_{\kappa}^{FN}(t) = \theta_1 * \ \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{FN}(t) + \theta_2 * \xi_{\kappa}^{FN}(t) \quad (8)$$

Cloud Computational Model: prominent for its limitless resource capabilities, the compute intensive high-end tasks are forwarded to cloud layer. The time spent while offloading task $\mathcal{T}_{\kappa}$ to cloud datacenter.

$$\mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{\mathcal{CD}}(t) = \frac{\mathcal{T}_{\kappa}(t)}{\omega_{\mathcal{CD}}} + TD_{WAN}(t) + \mathcal{PD}_{WAN}(t); \ \forall \mathcal{T}_{\kappa} \in \mathcal{T} \quad (9)$$

Where $\omega_{\mathcal{CD}}$ represents computational frequency of cloud datacenter and $\mathcal{TD}_{WAN}$ represents transmission delay of underlying network, whereas propagation delay of medium is demoted as $\mathcal{PD}_{WAN}$.

$$\text{where } TD_{WAN}(t) = \frac{\mathcal{T}_{\kappa}(t)}{ßW_{\kappa}^{FC}} \quad (10)$$

The EC at this layer while executing offloading decision is computed as:

$$\xi_{\kappa}^{\mathcal{CD}}(t) = \rho_{\kappa}^{ex \to \mathcal{CD}}(t) * \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{\mathcal{CD}}(t) \quad (11)$$

The total processing time at this layer is calculated as,

$$\mathbb{C}_{\kappa}^{\mathcal{CD}}(t) = \theta_1 * \ \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{\mathcal{CD}}(t) + \theta_2 * \xi_{\kappa}^{\mathcal{CD}}(t) \quad (12)$$

Finally, computation of overall cost to service task generated by device d, is computed based upon the offloading decision variable φ_k , and is represented as follows in the form of eq (13):

$$\mathbb{C}_{d\,Total}(t) = \sum_{k=1}^{n}(1 - \varphi_k^2(t))\ \mathbb{C}_{\kappa}^{IoT}(t) + \sum_{k=1}^{n}\varphi_k(t)(\varphi_k(t) - 1)\mathbb{C}_{\kappa}^{FN}(t) + \sum_{k=1}^{n}\varphi_k(t)(1 + \varphi_k(t))\ \mathbb{C}_{\kappa}^{\mathcal{CD}}(t) \quad (13)$$

In a similar manner, the total energy consumption from eq. (3), (7) and (11) can be combined using offloading decision variable as:

$$\xi_{d\,Total}(t) = \sum_{k=1}^{n}(1 - \varphi_k^2(t))\ \xi_{\kappa}^{IoT}(t) + \sum_{k=1}^{n}\varphi_k(t)(\varphi_k(t) - 1)\xi_{\kappa}^{FN}(t) + \sum_{k=1}^{n}\varphi_k(t)(1 + \varphi_k(t))\ \xi_{\kappa}^{\mathcal{CD}}(t) \quad (14)$$

Now, combining (13) and (14) results in formulating total cost corresponding to device d at time slot t, which is represented as:

$$\mathrm{C}_d(t) = \Phi_1 \mathbb{C}_{d_{Total}}(t) + \Phi_2 \xi_{d_{Total}}(t) \quad (15)$$

Here, Φ_1 and Φ_2 signifies weight parameters, which weighs processing cost and energy consumption respectively, in accordance with their weight factors.

C. PROBLEM FORMULATION

Makespan Time (MST): The aim of our work is to minimize the makespan time of IIoT requests, the total execution time at various layers corresponding to task $\mathcal{T}_{\kappa}$ and j[th] resource is computed as follows

$$\mathcal{E}\Gamma_{\text{Total}}^{\mathcal{T}_{\kappa},j} = \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{IoT}(t) + \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{FN}(t) + \mathcal{E}\Gamma_{\mathcal{T}_{\kappa}}^{\mathcal{CD}}(t) \quad (16)$$

Finally, maximum completion time corresponding to resource set $\mathfrak{R}_{\mathcal{M}}$, is computed as follows:

$$\text{MST}(\mathfrak{R}_{\mathcal{M}}) = max \sum_{k=1}^{n}\sum_{j=1}^{\mathcal{M}} \mathcal{E}\Gamma_{\text{Total}}^{\mathcal{T}_{\kappa},j} \quad (17)$$

To sum up with, we aim to establish a task offloading optimization function by formulating total cost minimization function, which is defined as follows:

$$\mathcal{F} = \min_{\varphi} \sum_{d=1}^{n} \varpi_{\text{d}}\, \mathrm{C}_{\text{d}}(t) \quad (18)$$

Where ϖ_d depicts the task priority corresponding to each device d to be considered for offloading and hence (18) validates the following constraints:

$$\varphi_k(t) \in \{-1, 0, 1\} \quad (19)$$

$$ßW_{\kappa}^{FC} \leq ßW_{\text{Max}}^{FC} \quad (20)$$

$$ßW_{\kappa}^{IF} \leq ßW_{\text{Max}}^{IF} \quad (21)$$

$$\omega_{IoT} < \omega_{FN} < \omega_{\mathcal{CD}} \quad (22)$$

$$\forall t \in T, \forall d \in \mathfrak{D}, k \in \mathcal{T} \quad (23)$$

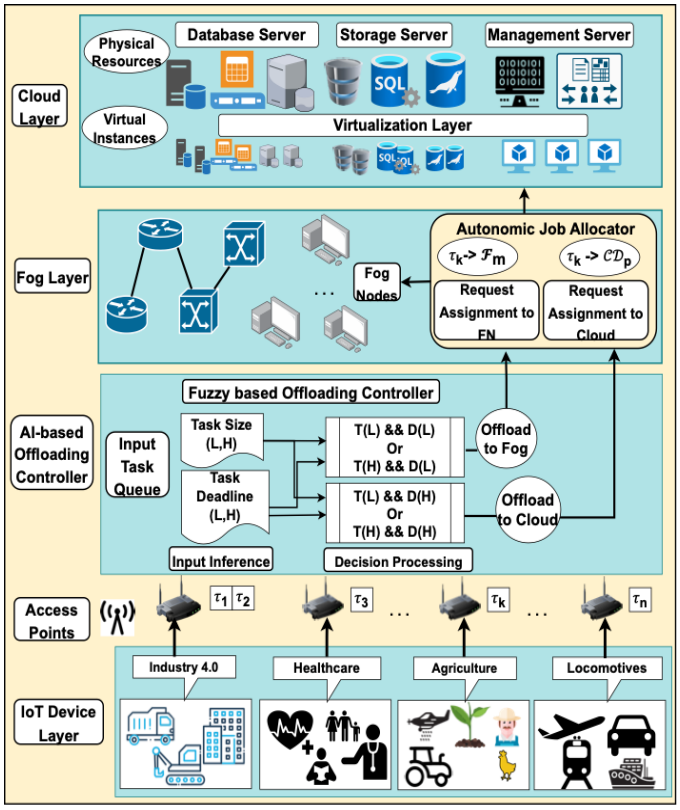

Fig 1: Proposed Resource Offloading and Allocation Framework in IIoT scenario

$$\Phi_1 + \Phi_2 = 1 ; 0 \leq \Phi_1 \leq 1; 0 \leq \Phi_2 \leq 1 \quad (24)$$

IV. PROPOSED FRAMEWORK FOR OFFLOADING DECISION AND RESOURCE ALLOCATION

Our proposed framework highlights the industry 5.0 scenario which is emerging with modern computing paradigms such as collaborative cloud-fog landscapes, which deliver a distributed environment comprising various resources scattered across consumer devices, FNs and cloud servers. The consumer devices constitute the IIoT layer, which covers a multitude of use cases such as Industry 5.0, smart healthcare, smart agriculture, automated locomotives, smart power, grids and so on, and generates requests for underlying resources in a collaborative cloud-fog environment. Furthermore, such applications thrive for prompt service delivery, which can't be fulfilled by edge devices. Hence, resource allocation becomes a predominant issue for optimizing the system's overall performance, ensuring sustainable usage of resources. This multi-layer architectural building block of our work has been put forward in the form of distinct layers.

1) *IIoT Layer*: It comprises sensors and actuators which collect the data in distinctive formats and rates, which are subsequently passed to upper layers via Access Points (APs), which can be located across various locations such as railway stations, airports, educational and healthcare institutions etc. These APs are connected to FNs and cloud instances via high-speed transmission channels.

2) *Fog Layer*: a distributed service responsible for imparting data processing, aggregation, processing, and analytics in close proximity to the task source. This layer is incorporated with the autonomic request allocation module, which maps and allocates requests to the apt fog node, $\mathcal{F}_n$. However, complex tasks can't be processed at this layer and hence need to be forwarded to resource-affluent cloud datacenters. Fig. 1 illustrates an AI-enabled autonomic framework for task offloading and request allocation in a multilayer fog cloud environment.

3) *Cloud Layer*: It offers limitless storage, processing, and networking capabilities in the form of database, storage, and management servers. The virtual instances of physical resources are accessible in the form of virtual instances known as VMs, that run in remote datacenters. In addition, a metaheuristic Job allocator module was proposed that optimally places the incoming job to the best possible FN (in the case of tasks offloaded to the fog layer) or cloud instance (in the case of tasks offloaded to cloud datacenter virtualized instances).

V. PROPOSED TASK OFFLOADING ALGORITHM

Herein, a task offloading algorithm along with its scheduling model are presented, incorporating AI-based techniques comprising fuzzy inference and metaheuristics.

A. FUZZY-BASED TASK OFFLOADING ALGORITHM

Each incoming tasks $\mathcal{T}_k$, where $\mathcal{T}_k \in \mathcal{T}$ contains a priority and a deadline associated with it. To accomplish the QoS objectives, it becomes critical to classify jobs according to their necessities and define the layers for offloading IoT requests. The proposed offloading module analyses the task perceived based on $\mathcal{T}_{\kappa}(t)$<High, Low> and $\mathcal{T}_{d}(t)$ <High, Low>. Accordingly, the task offloading destination is decided.

Our proposed algorithm categorizes each incoming task $\mathcal{T}_{k(t)}$ into two classes as class 1 (C1) and class 2 (C2), using fuzzy inference logic. This grouping of tasks allows their concurrent execution at distinct layers, avoiding ageing, and being preemptive in a multiprocessing environment [16]. All tasks with a long length and a tight deadline or a low length with a hard deadline fall into class, C1.

For processing, these jobs typically require more computationally intensive resources. As a result, cloud VMs handle this type of workload, because of the resource-intensive nodes, and reduce processing time. On the other hand, lower-value deadline tasks are considered best suited for forwarding to local servers due to their low latency.

Algorithm 1: Task Offloading algorithm for device d

Input: The total amount of tasks $\mathcal{T}_\kappa$ generated by device d
Output: Offloading decision $\varphi_\kappa(t)$

Initialize: Number of tasks(n); Number of Fog Nodes(m) ;Maximum working period of FN (T_{wp}); Current time (T_{cur})
while $T_{cur} < T_{wp}$ **do**
  for *t=1,2,3 …T* **do**
  Compute Total Computational Length; $\mathcal{T}_{\kappa_{Total}} = \sum_{k=1}^{n} \mathcal{T}_{\kappa(t)}$
  Compute Average computational Length; $\mathcal{T}_{\kappa_{Avg}} = \frac{\mathcal{T}_{\kappa_{Total}}}{m}$
  Compute Average task Deadline; $\mathcal{T}_{\kappa_{Avg_d}} = \frac{\sum_{k=1}^{n} \mathcal{T}_{d(t)}}{m}$
  for k = 1 to m do
  if ($\mathcal{T}_\kappa(t) \leq \mathcal{T}_{\kappa_{Avg}}$ *and* $\mathcal{T}_\kappa(t) \leq \mathcal{T}_{\kappa_{Avg_d}}$)
  Or ($\mathcal{T}_\kappa(t) > \mathcal{T}_{\kappa_{Avg}}$ *and* $\mathcal{T}_\kappa(t) \leq \mathcal{T}_{\kappa_{Avg_d}}$) **Then**
    Set $\varphi_\kappa(t)$= 1; Offload to Local Server (FN)
  end if
  else ($\mathcal{T}_\kappa(t) \leq \mathcal{T}_{\kappa_{Avg}}$ *and* $\mathcal{T}_\kappa(t) > \mathcal{T}_{\kappa_{Avg_d}}$)
  Or ($\mathcal{T}_\kappa(t) > \mathcal{T}_{\kappa_{Avg}}$ *and* $\mathcal{T}_\kappa(t) > \mathcal{T}_{\kappa_{Avg_d}}$) **Then**
    Set $\varphi_\kappa(t)$= -1; Offload to Cloud Server
  end for
  end for
end while

B. WOA-BASED RESOURCE ALLOCATION

After the offloading decision, it becomes significant to allocates best attainable resources for requests. We propose to exploit the functionality of a prominent metaheuristic technique known as the Whale Optimization Algorithm (WOA), an evolutionary and stochastic technique influenced by humpback whale hunting strategies used for the optimization of computationally hard problems for efficient resource allocation [7].

Encircling phase: It involves encircling the prey once the agent becomes location aware. Once the best candidate solution is assigned, other agents update their position, $\mathcal{D}$ as per the following equations [17]:

$$\mathcal{D} = |C\, \mathcal{X}^*(t) - \mathcal{X}(t)| \quad (25)$$

$$\mathcal{X}(t+1) = \mathcal{X}^*(t) - Å\, \mathcal{D} \quad (26)$$

Where $Å, C$ are coefficient vectors and $\mathcal{X}^*(t)$ represents position vector of best solution.

***Exploitation phase*:** It depicts the formulation for calculating the distance between whale location and its prey, and hence a helix-shaped movement of humpback is created as follows:

$$\mathcal{X}(t+1) = \mathcal{D}' \cdot e^{xy} \cdot cos(2\pi l) + \mathcal{X}^*(t) \quad (27)$$

Where $\mathcal{D}' = |\mathcal{X}^*(t) - \mathcal{X}(t)|$ (28)

And $\mathcal{Y} \in [-1,1]$ (29)

Bubble-net Attacking phase: It comprises of two mechanisms with which the humpback whales update their position as they swim around the prey. The mathematical equation of this phase is given as:

$$\mathcal{X}(t+1) = \begin{cases} \mathcal{X}^*(t) - Å \cdot \mathcal{D} & if\ \rho < 0.5 \\ \mathcal{D}' \cdot e^{xy} \cdot cos(2\pi l) + \mathcal{X}^*(t) & if\ \rho \geq 0.5 \end{cases} \quad (30)$$

Here, ρ represents a random number ranging [0,1]. Finally, α is utilized in order to measure the proximity of search agent, from reference whale.

$$\mathcal{D} = C * \mathcal{X}_{\Re}(t) - \mathcal{X} \quad (31)$$

$$\mathcal{X}(t+1) = \mathcal{X}_{\Re}(t) - \alpha * \mathcal{D} \quad (32)$$

Where constant, $C = 2 * \mathfrak{r}\ \forall\ \mathfrak{r} \in [0,1]$ (33)

And $\alpha < 1\ or\ \alpha > 1$ (34)

VI. PERFORMANCE EVALUATION

This section discusses the performance of the proposed AI based task offloading and incoming request allocation schemes for IoT devices that have been evaluated in dynamic cloud-fog environments.

Experimental Setup: The simulation experimentation scenario has been hosted on a system with an AMD Ryzen 5 5500U processor with Radeon Graphics, 2.10 GHz, Windows 11 x64 bit operating system, 256GB SSD and 12GB RAM. The impact of crucial QoS parameters of the proposed algorithm, including makespan, execution cost, task rejection and energy consumption are mainly investigated. We have considered a scenario where the number of tasks, $\mathcal{T}_\kappa(t)$ varies from 100-900 in range with a requirement for a distinct number of VMs in a multilayer architecture to service incoming IoT requests. In order to effectuate the proposed methodology, a synthetic dataset has been generated, comprising the task length and deadline for offloading decisions. Authors have set up a prescribed range for smooth and bursty i.e., 10,000 to 50,000 instructions per task with a deadline of 50 to 70 seconds. The dataset corresponding to VMs contains 2 parameters i.e., the execution speed of VMs and their cost. The range for VM speed varies between 1000-2000 MIPS and cost is dependent upon the computational capability of resources. The entire resource whether a fog device or a cloud datacenter, is heterogeneous in nature. When offloading the task to the cloud, transmission delay has been considered in the simulation experiment.

Algorithm 2: WOA-based Resource Allocation

Input: Population size ($\mathcal{Z}$), iterations($t_{\max}$),
Output: minimal cost function $\mathcal{f}$ and MST($\Re_{\mathcal{M}}$)

Initialize: $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots \mathcal{T}_m\}$, Maximum number of iterations ($t_{\max}$), position of $\mathcal{Z}$ whales randomly, coefficient vectors Å and C
for ($i = 1\ to\ i \leq \mathcal{Z}$) **do**
  Generate initial population $\mathcal{X}_i(t)$ randomly corresponding to i_{th} whale
  Compute the fitness function value $\mathcal{f}(\mathcal{X}_i(t))$ of each search agent
End for
While ($t \leq t_{\max}$) **do**
  For each $i = 1 : \mathcal{Z}$ **do**
  if (ρ < 0.5) **then**
    if (|Å| < 1) **then**
      Position of current search agent $\mathcal{X}_i(t)$ is updated using *Eq. (23)*
    else
      if (|Å|≥ 1) **then**
        Select a random search agent $\mathcal{X}_{\Re}(t)$
        Position of current search agent $\mathcal{X}_i(t)$ is updated using *Eq. (30)*
      End if
    End if
  else
    if (ρ ≥ 0.5) **then**
      Position of current search agent $\mathcal{X}_i(t)$ is updated using Eq. (25)
    End if
    Evaluate the fitness function of each search agent $\mathcal{f}(\mathcal{X}_i(t))$
  End if, End for
  best search agent $\mathcal{X}^*(t)$ is updated
Until ($t > t_{\max}$)
Return optimal value of MST($\Re_{\mathcal{M}}$), and $\mathcal{f}$

The proposed technique uses fuzzy logic-based intelligent decision making for task offloading and the whale optimization algorithm, written in Java for the allocation of resources to the upcoming IoT applications. To assess the proposed framework's performance, the author chose a diverse umbrella of approaches comprising static, dynamic and metaheuristics in the form of state-of-the-art work.

- *RR*: Round Robin algorithm is the most prominent static algorithm comprising procedural planning of CPU computing capabilities by incoming requests for fixed time quantum.
- *ROP* [8]: Random Offloading algorithm selects a random computing device on which the incoming task will be offloaded. This approach works in a dynamic manner without taking into consideration any objective parameters.
- *BAT* [9]: Nature-inspired evolutionary and stochastic techniques are making a noteworthy mark by optimizing computationally hard problems specifically in the arena of resource management for integrated cloud-fog landscape. This metaheuristic exploits characteristics of microbats such as echolocation, frequency tuning, and automated zooming to target their prey.

A. *Performance analysis for the Makespan parameter*

It is defined as the time required to execute entire IoT requests by computing platform (fog node or cloud server), which is depicted by equations (16) and (17). $\Phi_k(t)$ is a decision variable that determines whether a task is assigned to a given VM deployed on a local server or in the cloud. The authors have considered two test cases:

Case 1: Initially 100 tasks have been taken to evaluate the performance of the latency-aware proposed framework. The proposed WOA algorithm has the ability to search for the best possible resource for the execution of IoT requests with higher convergence accuracy, that leads to a global optimal solution. The WOA allocate the optimal resource to the tasks based on its requirements and compared it with state-of-the-art approaches to validate the results. Further, the number of tasks is varied from 100 to 300 and up to 900 to test the performance of the proposed approach over a bursty workload while keeping 25 fixed VMs. The computational results shown in Fig. 2 prove that the proposed WOA integrated with the framework allocates the best resources for the execution of IoT tasks in minimum time with high convergence speed.

Case 2: The authors have varied the number of resources from 10 to 50 while fixing the upcoming IoT tasks (500) to measure the efficiency of the proposed framework at makespan as a QoS parameter. Figure 3 depicts the superior performance of WOA as compared to its counter parts in collaborative cloud fog environment while considering deadlines as constraints for latency-sensitive applications. The experimental results of the proposed framework in different test cases reveal that the proposed scheme reduces the makespan time by 5.22% , 21.42%, and 37.17%, respectively, compared with the BAT, RR, and ROP schemes.

B. Performance Analysis for Task Rejection

It is the ratio of rejected tasks to the total number of submitted tasks. The mere reason for task rejection occurs due to the incompetence of underlying resources (a fog node or cloud datacenter) to execute the task within the given deadline.

$$T_{Rej} = \frac{Number\ of\ tasks\ rejected}{Total\ tasks} * 100 \quad (33)$$

In order to estimate the proposed algorithm's performance, the author has considered 2 scenarios. The former scenario

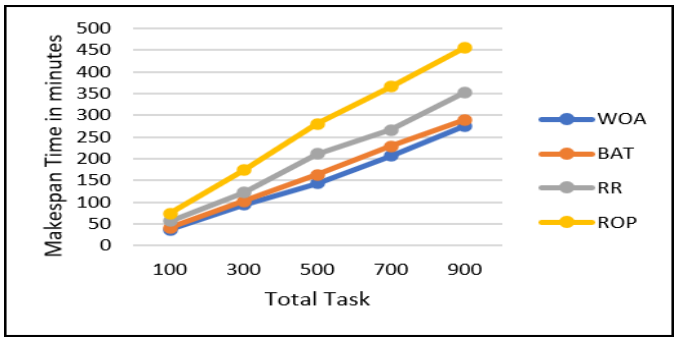

Fig 2: Makespan comparison with diverse tasks

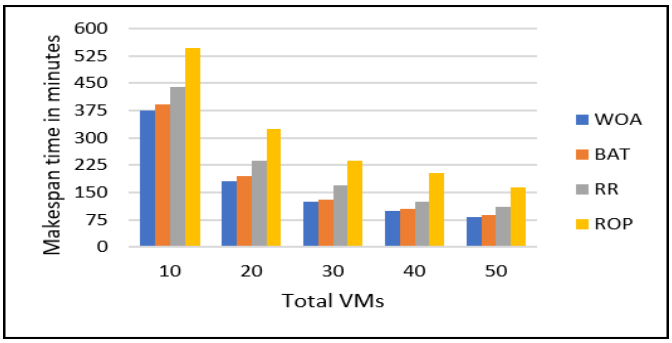

Fig 3: Makespan comparison with varying VMs

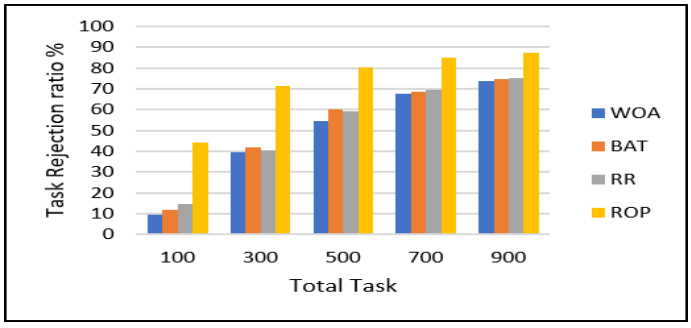

Fig 4: Illustrating Task Rejection Ratio with diverse tasks

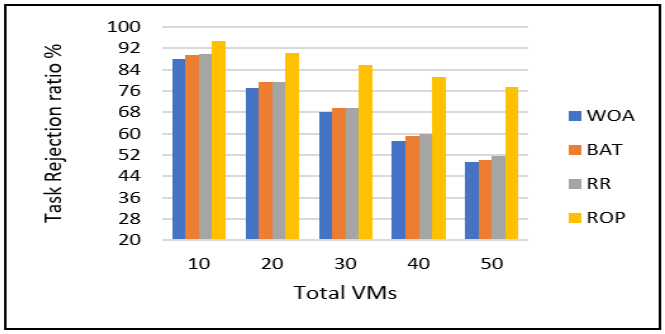

Fig 5: Illustrating Task Rejection Ratio with varying VM's

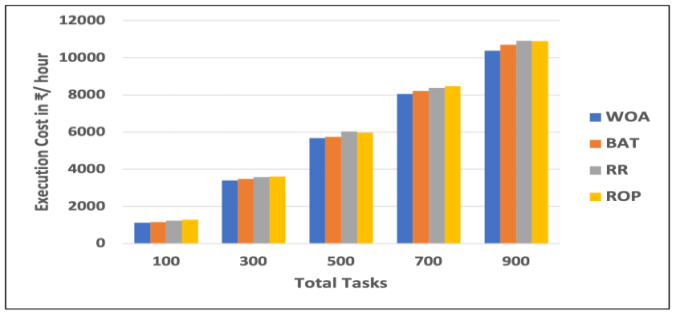

Fig 6: Evaluation of Execution cost with varying task length

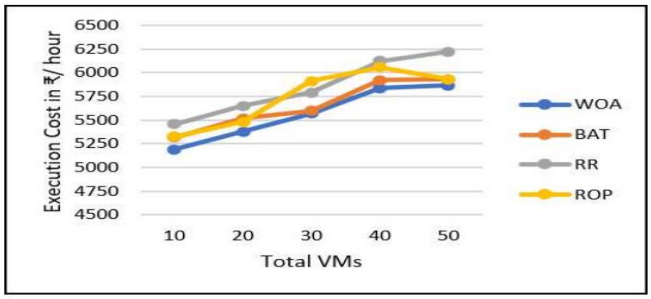

Fig 7: Evaluation of Execution cost with varying VMs

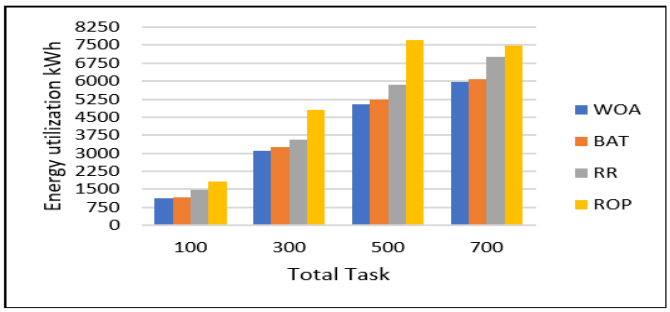

Fig 8 Analysis of Energy utilization with fixed VM's

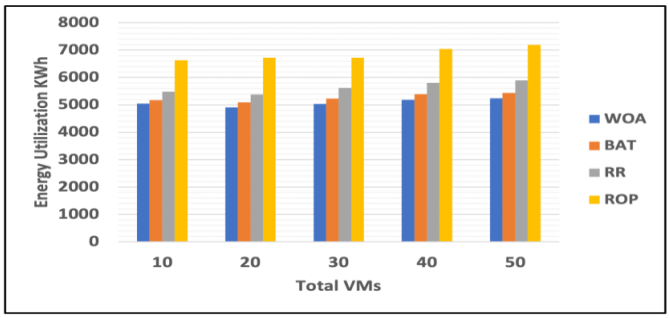

Fig 9: Analysis of energy utilization with fixed task length

Consider a varying number of tasks ranging from 100 to 900. Along with this, the number of VMs have been fixed at 25. These incoming tasks $\mathcal{T}_\kappa \in \mathcal{T}$ are processed via various state-of-the-art algorithms along with our proposed technique. As demonstrated in Fig 4, it is concluded that WOA serves the incoming tasks in a better manner satisfying the task constraints in terms of length $\mathcal{T}_\kappa(t)$ and deadline, $\mathcal{T}_d(t)$. The later scenario incorporates variation in terms of underlying resources (VMs in fog nodes or cloud datacenters) keeping the number of IIoT tasks fixed. The incorporation of fuzzy modules in the proposed technique shows astounding results providing optimal output and meeting the IIoT task deadline as depicted in Fig. 5.

C. Performance Analysis for Execution Cost

To serve the real-time IoT workload, the cost incurred by the user by the service provider remains a primary concern. Hence, the author has evaluated the proposed technique in

terms of execution cost, where the cost model has been portrayed in terms of Indian Rupee, ₹. The cost model has been formulated as a function of the sum total of execution time $\mathbb{C}_{d_{Total}}(t)$ and EC, $\xi_{d_{Total}}(t)$ from eq. (15). The test cases considered demonstrate improvised results proving that the proposed technique outshines other state-of-the-art algorithms by outlaying minimal execution cost incurred, in a fixed task length and fixed VM's scenario as shown in Fig 6 and 7 respectively.

D. Performance Analysis for Energy Utilization

Energy consumption remains a dominant parameter with the increased number of IoT requests and the increasing CO_2 footprint in the environment is a genuine concern [18]. Hence, the author evaluated the EC by devices and VM's at various layers formulated in eq. (3), (7) and (11) of the proposed algorithm against the benchmark approaches. Considering both cases, authors have observed that the proposed approach improved the EC up to 6% in the case of BAT, 27.32% in the case of ROP and 14% in comparison to RR. The results have been illustrated in Fig 8, which considers the case of fixed VMs and Fig 9, which portrays an energy matrix comparison with a fixed incoming task length.

VII. CONCLUSIONS AND FUTURE WORK

With the rising era of IIoT applications comprising an orchestrated cloud-fog computing paradigm, managing the finite resources, and ensuring prompt response delivery has become a challenging task. Hence, our work identifies and addresses the most challenging issues in offloading tasks and allocating requests by procuring an AI-enabled framework. The decision to offload tasks and request allocation has been formulated as an optimization problem. The decision of where to execute an Industry 5.0 incoming task is optimized using a fuzzy based model at access points considering deadline constraints and moreover boosting the battery life of the IIoT device. The framework is complemented with an autonomic nature-inspired evolutionary and stochastic approach well known as WOA, which holds prominence for devising optimal solutions for computationally hard problems in minimal time. The proposed computational model has been validated using extensive experiments in terms of vital QoS stats comprising makespan, task rejection, execution cost and energy consumption. Two scenarios have been considered, one with a fixed number of VMs and the other with a fixed number of tasks. Escalating results depict 2% and 3% reductions in makespan and task rejection ratios in contrast to BAT, ROP and RR techniques. For future work, we propose to incorporate machine learning techniques for workload prediction to further improve the offloading response to consumers and IIoT devices in Industry 5.0

REFERENCES

[1] F. Alqahtani, M. Amoon, and A. A. Nasr, "Reliable scheduling and load balancing for requests in cloud-fog computing," *Peer Peer Netw Appl*, vol. 14, no. 4, pp. 1905–1916, Jul. 2021, doi: 10.1007/S12083-021-01125-2/FIGURES/9.

[2] N. Kumari, A. Yadav, and P. K. Jana, "Task offloading in fog computing: A survey of algorithms and optimization techniques," *Computer Networks*, vol. 214, p. 109137, Sep. 2022, doi: 10.1016/J.COMNET.2022.109137.

[3] G. Carvalho, B. Cabral, V. Pereira, and J. Bernardino, "Computation offloading in Edge Computing environments using Artificial Intelligence techniques," *Eng Appl Artif Intell*, vol. 95, p. 103840, Oct. 2020, doi: 10.1016/J.ENGAPPAI.2020.103840.

[4] H. Liao *et al.*, "Learning-Based Context-Aware Resource Allocation for Edge-Computing-Empowered Industrial IoT," *IEEE Internet Things J*, vol. 7, no. 5, pp. 4260–4277, May 2020, doi: 10.1109/JIOT.2019.2963371.

[5] A. R. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, and M. Alikarar, "A smart home energy management system using IoT and big data analytics approach," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 4, pp. 426–434, Nov. 2017, doi: 10.1109/TCE.2017.015014.

[6] Y. Nan *et al.*, "Adaptive Energy-Aware Computation Offloading for Cloud of Things Systems," *IEEE Access*, vol. 5, pp. 23947–23957, Oct. 2017, doi: 10.1109/ACCESS.2017.2766165.

[7] C. Chakraborty, K. Mishra, S. K. Majhi, and H. K. Bhuyan, "Intelligent Latency-Aware Tasks Prioritization and Offloading Strategy in Distributed Fog-Cloud of Things," *IEEE Trans Industr Inform*, vol. 19, no. 2, pp. 2099–2106, Feb. 2023, doi: 10.1109/TII.2022.3173899.

[8] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A Deadline and Priority-Aware Task Offloading in Fog Computing Framework Leveraging Multilevel Feedback Queueing," *IEEE Internet Things J*, vol. 7, no. 7, pp. 5773–5782, Jul. 2020, doi: 10.1109/JIOT.2019.2946426.

[9] S. Goyal *et al.*, "An Optimized Framework for Energy-Resource Allocation in a Cloud Environment based on the Whale Optimization Algorithm," *Sensors 2021, Vol. 21, Page 1583*, vol. 21, no. 5, p. 1583, Feb. 2021, doi: 10.3390/S21051583.

[10] M. Kumar, A. Kishor, J. K. Samariya, and A. Y. Zomaya, "An Autonomic Workload Prediction and Resource Allocation Framework for Fog enabled Industrial IoT," *IEEE Internet Things J*, pp. 1–1, Jan. 2023, doi: 10.1109/JIOT.2023.3235107.

[11] A. N.; Alvi *et al.*, "Intelligent Task Offloading in Fog Computing Based Vehicular Networks," *Applied Sciences 2022, Vol. 12, Page 4521*, vol. 12, no. 9, p. 4521, Apr. 2022, doi: 10.3390/APP12094521.

[12] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-Aware Marine Predators Algorithm for Task Scheduling in IoT-Based Fog Computing Applications," *IEEE Trans Industr Inform*, vol. 17, no. 7, pp. 5068–5076, 2021, doi: 10.1109/TII.2020.3001067.

[13] C. Liu and K. Liu, "Toward Reliable DNN-based Task Partitioning and Offloading in Vehicular Edge Computing," *IEEE Transactions on Consumer Electronics*, 2023, doi: 10.1109/TCE.2023.3280484.

[14] X. Wang and Y. Wu, "Fog-Assisted Internet of Medical Things for Smart Healthcare," *IEEE Transactions on Consumer Electronics*, 2023, doi: 10.1109/TCE.2023.3253896.

[15] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," *IEEE Internet Things J*, vol. 9, no. 1, pp. 1–24, Jan. 2022, doi: 10.1109/JIOT.2021.3095077.

[16] S. D. Okegbile, B. T. Maharaj, and A. S. Alfa, "A Multi-User Tasks Offloading Scheme for Integrated Edge-Fog-Cloud Computing Environments," *IEEE Trans Veh Technol*, vol. 71, no. 7, pp. 7487–7502, Jul. 2022, doi: 10.1109/TVT.2022.3167892.

[17] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, May 2016, doi: 10.1016/J.ADVENGSOFT.2016.01.008.

[18] M. Adhikari and H. Gianey, "Energy efficient offloading strategy in fog-cloud environment for IoT applications," *Internet of Things*, vol. 6, p. 100053, Jun. 2019, doi: 10.1016/J.IOT.2019.100053.