Minimax Theory and its Applications

Volume 11 (2026), No. 4s, 253 to 273

A Dynamic Ripple-Driven Adaptive Federated Swarm Optimization Framework for Scalable and Energy-Efficient Resource Management in IoT-MEC Networks

M.Velvizhi[1*], Dr.D.Saraswady[2]

[1]Department of Electronics and Communication Engineering, Puducherry Technological University, Puducherry. Email :velvizhi.m@ptuniv.edu.in
[2]Department of Electronics and Communication Engineering, Puducherry Technological University, Puducherry. Email :dsaraswady@ptuniv.edu.in

Abstract
The Internet of Things (IoT) has expanded rapidly, generating massive data and computation demands that challenge traditional centralized cloud systems. Mobile Edge Computing (MEC) addresses these issues by processing data closer to devices, yet efficient resource management remains difficult due to dynamic workloads, energy constraints, and large-scale node diversity. Existing optimization methods are mostly static and centralized, leading to poor adaptability, unbalanced energy use, and scalability issues. To overcome these limitations, this research introduces a Dynamic Ripple-driven Adaptive Federated Swarm Optimization (DRAFSO)framework for scalable and energy-efficient resource management in IoT-MEC networks. The proposed DRAFSO framework integrates a ripple-inspired dynamic adaptation mechanism that emulates wave propagation to enable continuous, context-aware adjustments under varying network conditions. Combined with Federated Swarm Intelligence, the system promotes decentralized optimization and collaborative decision-making among MEC nodes without sharing raw data, thus enhancing both privacy and scalability. DRAFSO jointly optimizes clustering, routing, and computation offloading through ripple-driven adaptive updates, ensuring balanced resource allocation and minimal latency. The primary objective is to design a dynamic, scalable, and energy-efficient optimization strategy that enhances adaptability and system lifetime while reducing energy consumption. Experimental results show that DRAFSO significantly improves convergence speed, energy efficiency, and network robustness compared to traditional algorithms. By enabling real-time adaptability and federated cooperation, the framework ensures sustainable and intelligent resource management across heterogeneous IoT-MEC environments. This innovation establishes a new direction for next-generation edge networks, achieving optimal performance through dynamic, privacy-preserving, and energy-aware optimization.

Keywords: Dynamic Ripple Mechanism, Federated Swarm Optimization, IoT-MEC Networks, Energy Efficiency, Resource Management, Adaptive Clustering, Computation Offloading, Load Balancing, Scalability, Edge Intelligence

1. Introduction

Cloud computing needs the effective offloading of computationally intensive tasks on power-constrained mobile phones and tablets to the massively powerful Cloud Server (CS). Users are allowed to offload a part of local data to an edge server and simultaneously train statistical models across several mobile devices. The special focus on minimizing the consumption of computation and communication energy should be given to a power-efficient or green FL system [1]. Users can also use smaller subsets of their local datasets, and training can decrease the performance of the model due to the energy constraints of the handheld devices [2]. The idea of using the MEC infrastructure to distribute the computing resources of resource-constrained mobile-based devices when training the FL is thus quite viable. The global structure can be sustained by enabling users to transfer some of the local data to edge servers and reducing the amount of energy used by the handheld devices [3]. Figure 1 shows a three-layer architecture that decreases the network traffic and stresses the cloud server by applying applications of the IoT devices to the fog layer, rather than forwarding them to the cloud. In case the resources of the fog devices are not enough to fulfil some of the tasks, the workload is transmitted to the cloud server [4]. Users would like to have tasks done as fast as possible to reduce the time of response, minimize the

associated costs, and achieve efficiency. The service providers are interested in minimal electricity use by computing devices to minimize carbon footprints and maximize profits [5].

Even though the operational goals of the fog and cloud nodes are different, they need to be energy efficient. Fog nodes are usually powered by batteries or other restricted power sources; therefore, better energy usage will increase the life of the node. The use of cloud nodes actually results in low unit energy costs and carbon emissions because of the enormity of the servers [6]. The use of intelligent capabilities is utilized by Edge Nodes (ENs) to reduce network loads at the core network by minimizing power consumption and supporting time-sensitive IoT services like interaction, data storage, and data analysis [7]. This is by updating local data and sending it to ENs, and at the same time getting the global model parameters sent by the ENs. FL is dependent on the aggregated local updates of the global model. The training models are refined by operating all IoT devices through updating global and local parameters until they achieve the required level of accuracy [8].

Edge computing and fog computing have been proposed to alleviate these problems. MEC is a new solution that allows the Wireless Devices (WDs) to connect with cloud-based applications through Base Stations (BS) and Access Points (AP), which have edge processors. Resource-constrained WDs have the capability of offloading computational workloads to APs within the MEC system and implementing the workloads using MEC servers [9]. In contrast to cloud computing, edge computing puts computing resources near endpoint devices, at the network edge. This closeness enhances the bandwidth of communication with remote data centers and the overall communication latency. New data sets are possible, based on the ease with which users can process and store information. Consequently, computation offloading in MEC has emerged as a key concern of the present-day study [10].

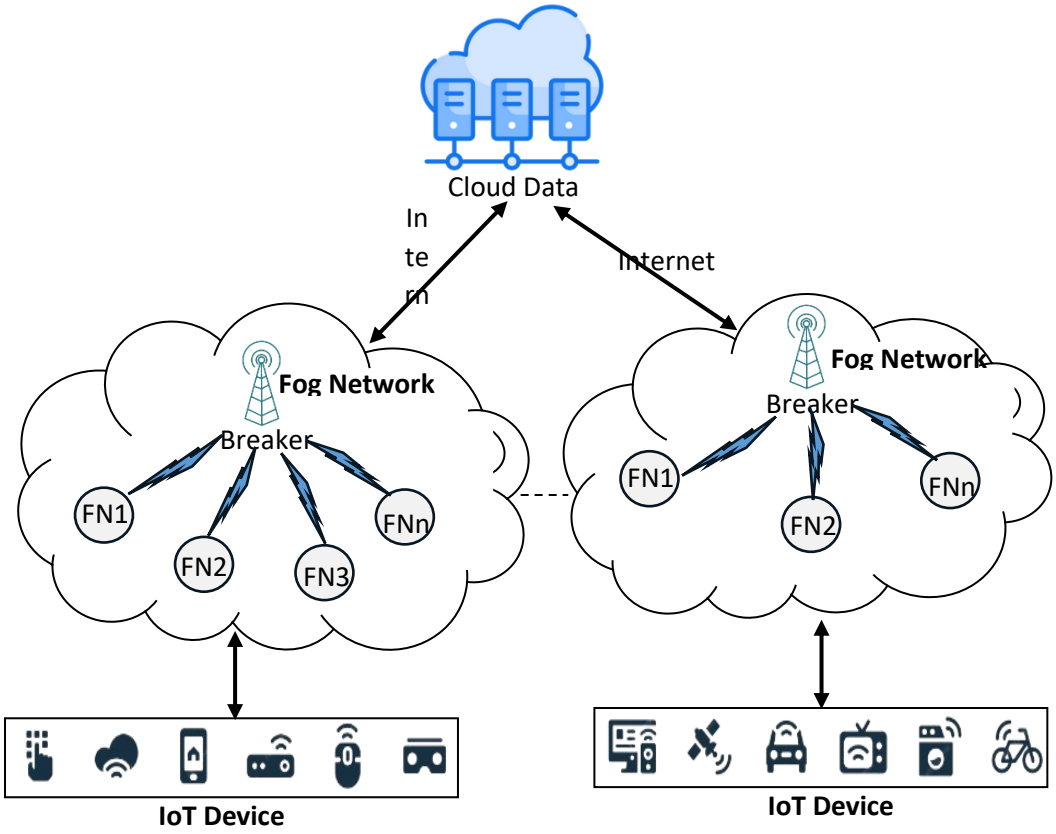


Fig.1 3 Layer architecture of Cloud with IoT device

Beyond transforming various industries, the IoT has created a need among organizations for information management platforms with the flexibility and high performance to handle the large volumes of data produced by these devices. Real-time applications and data processing depend heavily on cloud computing because they rely on large-scale deployments, but they are often affected by latency, bandwidth, and high energy consumption [11]. To identify a possible solution to these connectivity issues, edge computing could be a potential answer by processing the data where it is generated, reducing the latency of information transfer, and contributing to a decrease in the congestion of the network. Recent demands for sustainable energy solutions introduce further challenges to edge computing systems, particularly as the use of IoT-related applications is

growing [12]. Energy efficiency is a very important issue in computers because of the operational cost and the environmental issue. The energy consumption of IoT devices and edge nodes is high, forcing organizations to create new strategies that can get the most out of the energy and achieve high computational rates [13]. Key contributions of the paper are as follows:

- Introduced a dynamic ripple-driven mechanism for real-time adaptive resource optimization.
- Integrated federated swarm intelligence ensuring decentralized, privacy-preserving collaborative decision-making.
- Enhanced energy efficiency and scalability across heterogeneous IoT-MEC network environments.
- Unified clustering, routing, and computation offloading within a single optimization framework.
- Achieved faster convergence, reduced latency, and improved network lifetime over existing methods

2. Related Works

In recent times, researchers have shown great interest in coming up with energy-efficient edge computing architectures in IoT applications. The nature of IoT-based systems is dynamic and is associated with the limitations of the system structure, since it may result in workload variations and changing operating conditions. The need to satisfy the real-time demands of the applications is behind the increased demand to have systems that can do adaptive resource allocation [14]. With machine learning methods being used to make edge computing more sustainable, the networks will be able to optimize their energy consumption by using AI-oriented predictive processes. The important consideration when developing energy-efficient edge computing platforms is long-term scalability. Distributed IoT applications are deployed in decentralized environments where the number of devices as well as the volumes of data is growing at a very high rate [15]. Existing edge architectures might find it difficult to be energy efficient and high-performance as the number of devices to be supported keeps increasing. To tackle this issue, distributed modular computing edge designs that can be extended as the number of IoT demands grows with minimum energy consumption, have been the center of recent studies [16].

These edge computing systems save energy usage by multiple nodes using resource-sharing approaches and balancing workloads on distributed computing platforms. The strategy to utilize power-efficient CPUs with low-power communication protocols is needed to design energy-efficient computational edge systems [17]. Such low-energy hardware in remote or energy-limited settings is very important in ensuring low total power use. Although appropriate hardware technologies are at hand, computerization of intelligent software that can effectively control these resources is not a very easy job. Resource allocation algorithms that are oriented towards learning are critical in realizing optimal performance [18]. Concentrated on the distribution delay to save resources and not to make more than necessary computations to ensure high accuracy in delay modelling FL. The convergence constraints and policy scheduler design based on it directly influence the convergence rate of the trained system and minimize the FL loss. The computational constraints of IoT devices due to energy and processing constraints, tackling the varying amount of data with changing time, have a major limitation to the suitability of FL across internet links [19].

The ability to adjust the rate of operation of the Central Processing Unit (CPU) to another level of frequency permits regulating the rate at which the CPU will run its operations and therefore control the overall energy usage. Several researchers have also studied the allocation of tasks to different tasks within IoT systems using RL to reduce the time to complete the task [20]. To respond dynamically to changes in wireless channels and storage capacity, an RL-based approach was proposed in order to shorten the job completion latency without explicitly taking into account the limitations of the device battery. Other researchers have investigated the means of minimizing the delay of IoT services by using cooperative strategies and offloading policies. Whale Optimization Algorithm (WOA) has received a lot of focus due to its ease and effectiveness [21]. In spite of these merits, WOA has poor exploring power. It is basically an exploitation process, and it usually stops short at local optimum solutions instead of the global optimum solution. This constraint decreases the rate of convergence of WOA. A good timing of the tasks in WOA demands a little change in every iteration, but with time, the behaviour decreases its capabilities of converting poor agents into an optimal agent [22].

To overcome these issues, the proposal has been Whale Optimization Algorithm (RWOA), which is Ripple-induced to enhance task scheduling in the heterogeneous resources of cloud computing. To promote exploration, the RWOA augments ripple-based perturbation, thus avoiding untimely convergence and guiding improved global optimization capabilities. RWOA boosts diversity and faster convergence rate of the standard WOA by ripple effecting the fitness values of the two weakest agents in the search space. The resulting ripple effects enhance the number of solutions with larger fitness, hence enhancing the likelihood of having the least promising agents develop to become competitive candidates.

This improvement is attained by three different ripple mechanisms: forward, reverse, and bidirectional. Various computational problems have different requirements and constraints, including memory, bandwidth, and execution time. It is important to choose the right computing node in a fog cloud system to meet these limitations and to improve the QoS. Task scheduling in this type of environment is an optimization problem because it has to accommodate various criteria and constraints at the same time. This work aims to maximize network coverage and minimize energy consumption, as the multi-objective NP-hard optimization problem cannot be efficiently solved using existing approaches within a reasonable time frame.

3. Problem Formation

Consider an IoT-MEC network with N IoT devices and M edge servers (MEC nodes). Each device $x \in \{1,2,\dots,N\}$ generates a task T_x characterized by computational workload C_x in CPU cycles) and data size D_x in bits). The goal is to minimize total energy consumption while ensuring latency constraints and balanced resource allocation across MEC nodes.

Latency Model: The total task execution latency L_x for device x can be expressed as:

$L_x = L_x^{ti} + L_x^{comp} + L_x^{ri}$ (1)

Where:$L_x^{ti} = \frac{D_x}{R_{x,y}}$the transmission delay to MEC node y, with $R_{x,y}$ as the uplink data rate.$L_x^{comp} = \frac{C_x}{f_x}$is the computation delay at MEC node y, with f_x as allocated CPU frequency.$L_x^{ri}$is the result download time from MEC node to device.

The latency constraint is:$L_x \le L_x^{max},\ \forall x$

Energy Consumption Model: Energy consumed for offloading a task T_x is:

$E_x = E_x^{ti} + E_x^{comp}$ (2)

Where:$E_x^{ti} = P_x^{ti}.L_x^{ti}$, with P_x^{ti}as the transmission power.$E_x^{comp} = k.C_x.f_y^2$, where k is the computation energy coefficient.

The objective is to minimize total energy:$\min \sum_{x=1}^{N} E_x$

3.1 Constraints

Resource Allocation Constraints: CPU allocation per MEC node:$\sum_{x \in S_y} f_{x,y} \le f_y^{max},\ \forall y$

Where,S_y the set of tasks assigned to MEC node y.

Task assignment constraint:$i_{x,y} \in \{0,1\} \sum_{y=1}^{M} i_{x,y} = 1,\ \forall x$

Where,$i_{x,y} = 1$ if task x is offloaded to MEC node y.

3.2 Problem Statement

The optimization problem can be expressed as:$\min_{i,f} \sum_{x=1}^{N} \left(P_x^{ti} \frac{D_x}{R_{x,y}} + kC_x f_y^2 \right)$

$$s.t.\ L_x^{ti} + L_x^{comp} + L_x^{ri} \le L_x^{max}, \qquad \forall x$$

$$\sum_{x \in S_y} f_{x,y} \le f_y^{max}, \qquad \forall y$$

$$i_{xy} \in \{0,1\}, \qquad \sum_{y=1}^{M} i_{xy} = 1, \qquad \forall x$$

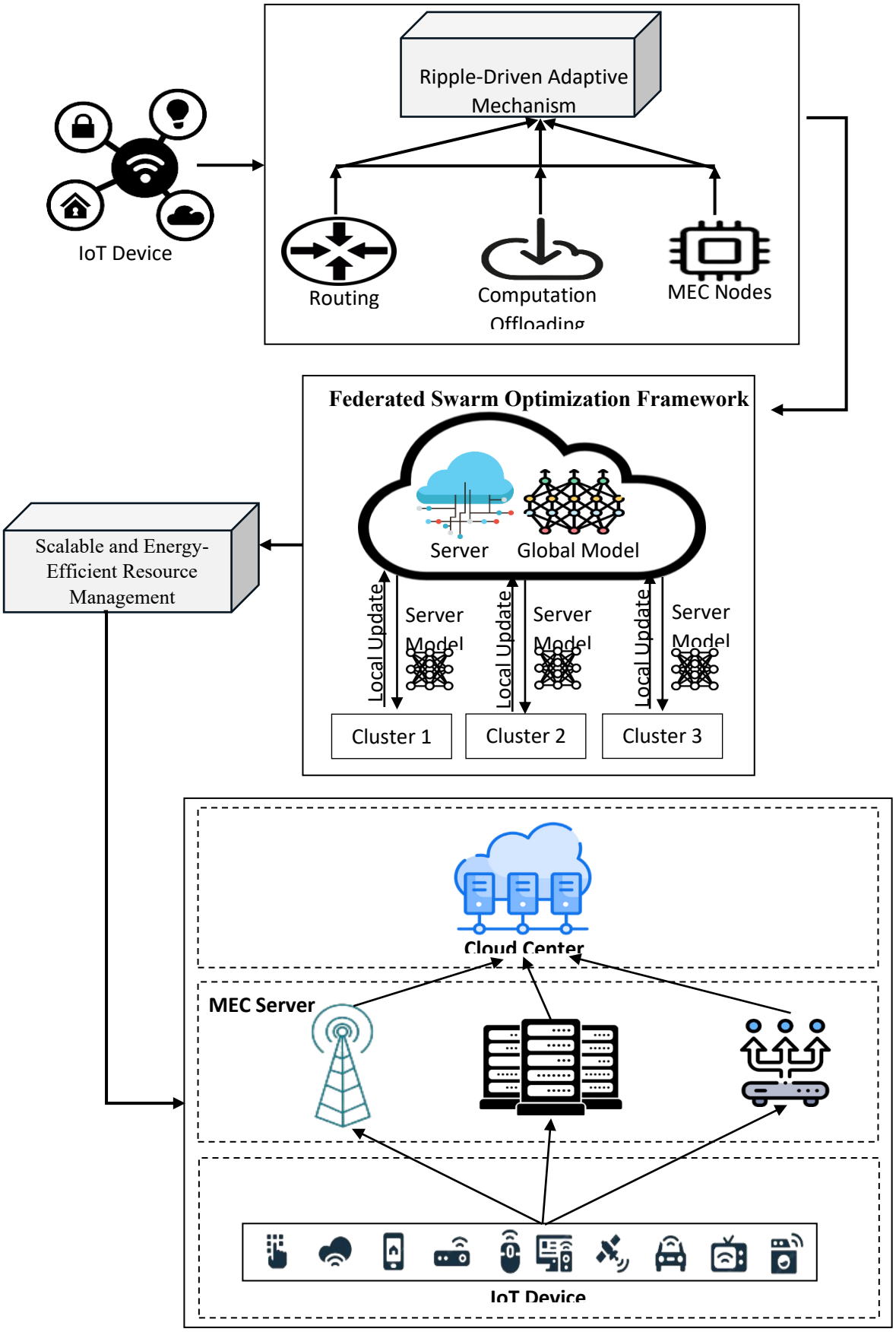


Fig.2 Proposed Architecture

This is a Mixed-Integer Non-Linear Programming (MINLP) problem that is NP-hard and requires Dynamic Ripple Adaptive Federated Swarm-based Optimization (DRAFSO) for efficient solution in large-scale IoT-MEC networks.

4. Proposed Methodology

A simulated IoT–MEC ecosystem comprising multiple IoT devices and edge servers with varying computational and energy capacities was used to test and evaluate the proposed DRAFSO architecture shown in Fig 2. The computational costs and data volumes of the tasks generated by each IoT device were continuously offloaded to neighboring MEC nodes. The system combined a ripple-inspired adaptive mechanism through which real-time cluster formation, task allocation, and route adjustments were attained by simulating the spreading of natural wave patterns. Cooperative efficiency was also improved through Federated Swarm Optimization (FSO), which maintained local data privacy through the coordination of decentralized decision-making between MEC nodes. The delay of task completion and energy usage was calculated using well-established models of computing and communication based on network data rates, transmission energy, and CPU frequency allocation. The approach proposed uses adaptive clustering, intelligent computation offloading, and optimization through iteration based on ripple to provide energy-efficient and environmentally friendly resource allocation.

4.1 Dataset description

The data used to test the proposed DRAFSO system consists of heterogeneous IoT devices and MEC nodes with computing, energy, and communication capabilities that are varied as indicated in Table 1. It simulates the real-world operating conditions with differences in the rate at which data are generated, maximum energy output, and the type of device (sensors, actuators, recorders, and smart devices). Network parameters such as bandwidth and latency, along with workload parameters, task size, and type, are used to analyse energy consumption and resource utilization. The dataset allows thorough testing of DRAFSO adaptability, scalability, and energy efficiency in dynamic and large-scale IoT-MEC systems by simulating both component and network heterogeneity.

Table 1. Dataset Description

Parameter	**Description**	**Range / Values**
Number of IoT Devices	Total connected IoT devices in the network	100 – 10,000
Device Type	Type of IoT device (sensor, actuator, camera, etc.)	Sensor, Actuator, Camera, Smart Appliance
Data Generation Rate	Amount of data generated per device per second	0.1 – 5 MB/s
MEC Node Count	Number of edge computing nodes deployed	5 – 50
CPU Capacity	Processing capacity of MEC nodes	2 – 16
Memory	RAM available per MEC node	4 – 64
Energy Capacity	Battery or energy limit of IoT devices	1000 – 10000 mAh
Workload Type	Type of computational task generated	Sensing, Video Processing, Analytics
Task Size	Computation size of each task	1 – 500 MB / 10 – 200 GFLOPS
Communication Latency	Time taken to transmit data between the IoT device and the MEC node	1 – 50 ms
Network Bandwidth	Available data rate for each link	10 – 1000 Mbps

Table 2. Sample Data

Device	**Device**	**Data**	**Battery**	**Task Type**	**Task**	**MEC**	**MEC**	**MEC**	**Latency**	**Bandwidth**

ID	Type	Rate (MB/s)	(mAh)		Size (MB)	Node ID	CPU (GHz)	RAM (GB)	(ms)	(Mbps)
D1	Sensor	0.5	2000	Sensing	5	M1	4	16	10	100
D2	Camera	2.0	5000	Video Processing	50	M2	8	32	20	200
D3	Actuator	0.2	1500	Analytics	10	M1	4	16	12	100
D4	Smart Appliance	1.0	3000	Sensing	20	M3	12	64	15	500
D5	Sensor	0.8	2500	Analytics	15	M2	8	32	18	200

This sample data illustrates a heterogeneous IoT-MEC ecosystem, as in Table 2. To have a real operational diversity, the devices are different in type, data generation rate, and battery capacity. The tasks are of an analytic, video processing, or sensing nature and vary in terms of format and computation needs. The hardware resources in the form of CPU, memory bandwidth, and network latency allocated to each hardware part affect the allocation of tasks to a particular MEC node. The data set allows simulating the load balancing, delay, and energy consumption vital measures to determine the performance of the DRAFSO system.

4.2 Ripple-driven adaptive mechanism with IoT-MEC networks

Ripple-Driven Adaptive Mechanism (RDAM) is based on the principle of natural waves or ripples propagation of a medium. Ripples produced by each edge node in the IoT-MEC network have information of its present workload, resource demands and available energy as illustrated in Fig 3. The propagation of these ripples through neighbouring nodes creates a dynamic influence field that determines dynamic allocation of resources. The size and frequency of the ripples are the volume and urgency of resource demands, and the fact that the intensity of the ripples reduces with time or distance with respect to distance describes how the impacts of the demands fade.

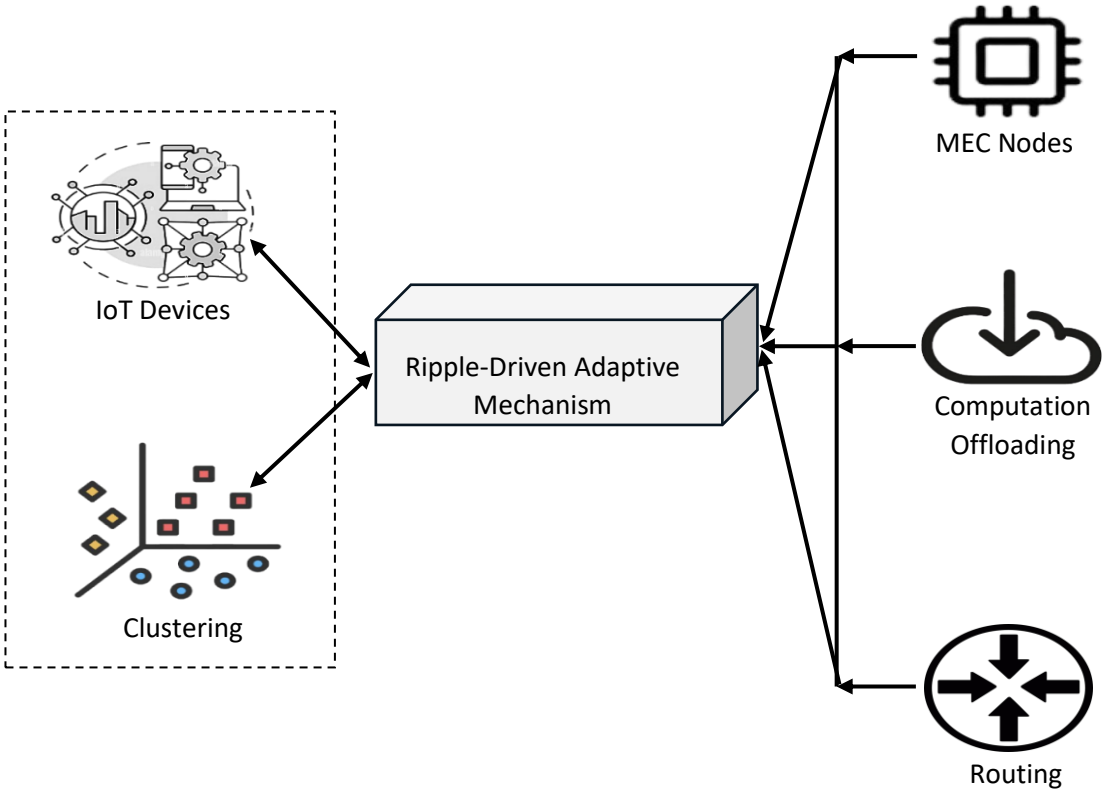


Fig. 3 Ripple-driven adaptive mechanism with IoT-MEC networks

The wave function for a ripple generated by a node x can be modeled as:

$R_x(t,d) = A_x.e^{-\lambda d}.\sin(\omega t + \emptyset_x)$ (3)

Where:$R_x(t,d) =$ = ripple effect from node i at time & and distance d; A_x = amplitude, proportional to the node's workload or resource demand; λ = decay factor controlling influence over distance; ω = frequency, reflecting urgency of task completion; $\emptyset_x =$ phase offset for synchronization with other nodes.

The adaptive allocation of resources is then determined by the superposition of ripples from all neighbouring nodes:$R_{total}(t) = \sum_{x=1}^{N} R_x\,(t, d_x)$ (4)

Where, N is the number of neighbouring nodes influencing a given node. The total ripple $R_{total}(t)$ defines the priority and amount of resources allocated to tasks dynamically: higher ripple amplitudes trigger more computation or bandwidth assignment.To incorporate energy efficiency, each node's ripple amplitude can be modulated by its residual energy E_x:

$A_x = \alpha.\frac{W_x}{E_x}$ (4)

Where:W_x = workload of node; E_x = residual energy of node x; α = constant

In this way, any high-workload, low-energy node will produce smaller ripples that will not allow overconsumption and will guarantee the allocation of resources in an adaptive manner that is energy conscious. RDAM uses the network as a dynamic ripple field, where the resource distribution is a natural phenomenon of the waves interaction and provides a scalable and energy-efficient performance on IoT-MEC environments.

4.3 Federated Swarm Optimization Framework for Scalable and Energy-Efficient Resource Management

This will reduce the internet usage costs and security risks, as it will collect samples of device data and will conduct learning locally. The basic concept of FSO framework of scalable and energy efficient resource management in the IoT-MEC networks is that the computational nodes are linked in a heterogeneous IoT setting as shown in Fig 4. Raw operational data collected by IoT devices are locally processed by several client nodes by collaborative learning and clustering of neighbouring MEC nodes. The structure also makes sure that the communication between nodes is exclusively through edge AI nodes and that, during training, the model parameters are kept constant, and as a result, this can improve the precision of training. To optimize overall system performance, the FSO framework aggregates global models across edge clusters and iteratively refines regional representations, achieving efficient scalability, privacy preservation, and reduced energy consumption. Swarm intelligence (SI) and FL concepts are combined in the FSO architecture to provide administration of resources that is flexible, scalable, and energy-efficient over dispersed IoT-MEC networks. To provide confidentiality-preserving coordination, each MEC node functions as an independent agent (or particle) that learns and improves its local resource distribution policies using local information along with cooperative updates from nearby nodes.The optimizing goal in this structure is to optimize compute effectiveness across all nodes and decrease overall consumption of energy while adhering to service latency limitations. One way to define the worldwide function of objectives is as follows:

In the IoT-device local learning scenario, each device $n \in N$ performs T_x local iterations and uploads its local parameters ω_n (size d_n) to its serving F-AP k on RRB z. The number of CPU cycles to process one sample is C_n and the device CPU frequency is f_n cycles/sec), hence the computation time for one local Iteration $\frac{C_n D_n}{f_n}$is and the total local computation time required to reach local accuracy $\epsilon_l$is

$T_n^{(c)} = T_t.\frac{C_n D_n}{f_n}$ (5)

$T_t = \frac{2}{(2-\delta\beta)\delta\vartheta}\ln(1/\epsilon_l)$ (6)

Where, δ, ϑ, β constants. After local training, the transmission time to upload parameters from IoT device n to F-AP k on RRB z is

$T_n^{(w)} = \frac{d_n}{R_{n,k,z}^{up}}$ (7)

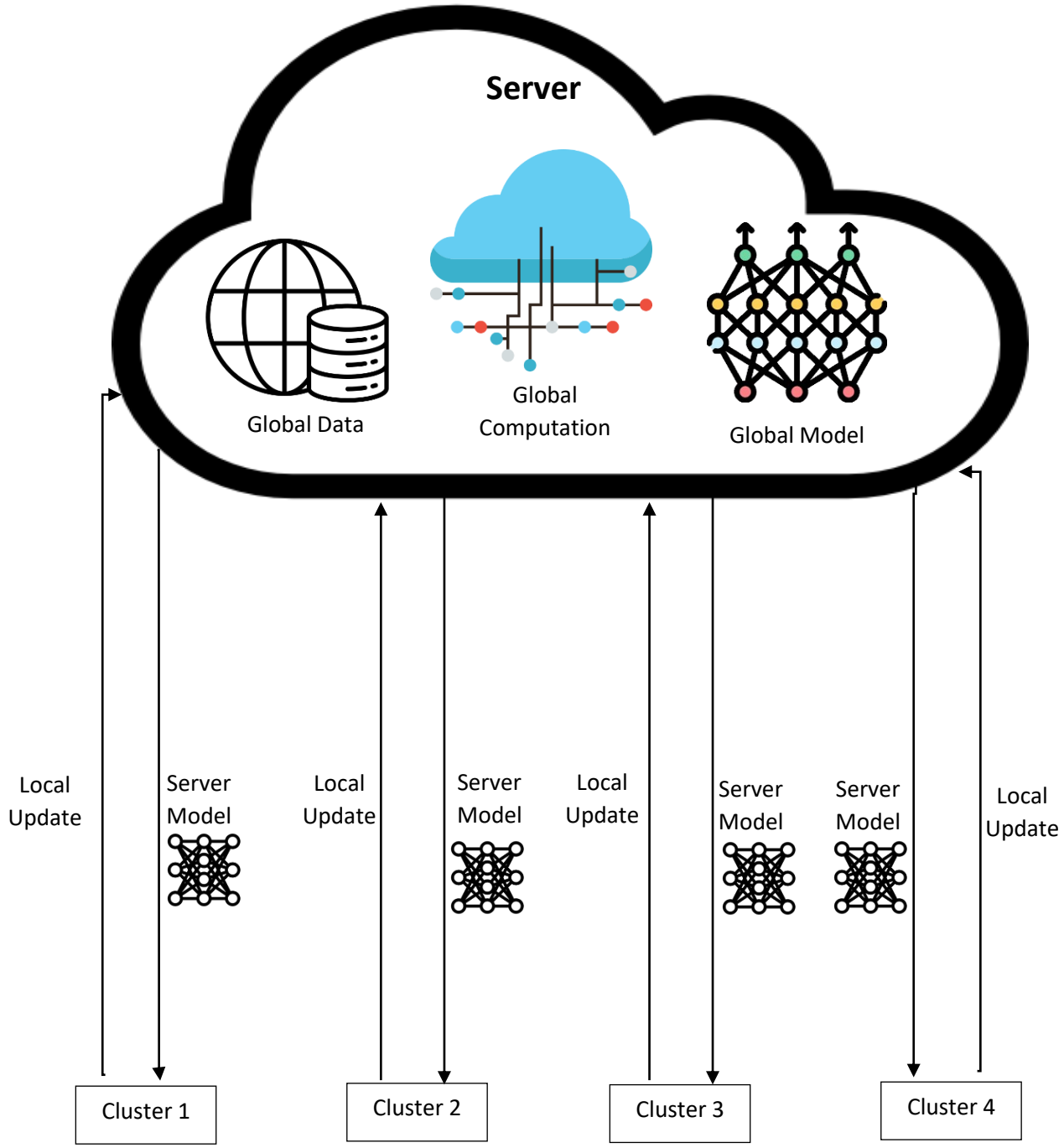


Fig.4 Federated Swarm Optimization Framework

And the time for F-AP k to forward its collected parameters to the CS is $T_n^{(w)} = \frac{\sum_{n \in N_k} d_n}{R_k^{fh}}$ (8)

Where,$R_{n,k,z}^{up}$ is the uplink rate of device n on RRB to F-AP k, R_k^{fh} the fronthaul rate of F-AP k to the CS, and N_k the set of devices served by F-AP k. The global model update occurs only after all local parameters are received and forwarded, the per-round federated learning time τ_1 is$T_l \frac{C_n D_n}{f_n} + \frac{d_n}{R_{n,k,z}^{up}} + \frac{\sum_{n \in N_k} d_n}{R_k^{fh}} \leq T_q, \ \forall n \in N, k \in K$

To capture energy costs, the computation energy for device n across T_l iterations is

$E_n^{(c)} = k_n C_n D_n f_n^2 T_l \ (J)$ (9)

Where k_n is the effective switched capacitance, and the transmission energy is

$E_n^{(t)} = p_n^{ti}.T_n^{(w)} = p_n^{ti}\frac{d_n}{R_{n,k,z}^u}$ (10)

So, the device energy budget constraint is $E_n^{(c)} + E_n^{(t)} \leq E_n^{max}$

Integrating swarm optimization and ripple adaptation, each IoT device is also a particle with allocation/strategy vector i_n (eg, choice of f_n RRB z, transmit power p_n^{ti}, local batch size). The DR-PSO velocity update for particle n at iteration t augmented by the cumulative ripple field $R_n^{total}(t)$ becomes

$v_n^{(t+1)} = \omega v_n^{(t)} + c_1 r_1\left(p_n^{best} - i_n^{(t)}\right) + c_2 r_2\left(g^{best} - i_n^{(t)}\right) + \gamma R_n^{total}(t)$ (11)

$i_n^{(t+1)} = i_n^{(t)} + v_n^{(t+1)}$ (12)

with $R_n^{total}(t) = \sum_{x\in N} A_x e^{-\lambda d_{ai}} \sin(\omega_r t + \phi_x)$ (13)

As in the RDAM model (amplitude A_x decay λ, ripple frequency ω_r). The swarm objective at device n minimizes a local cost combining time and energy while satisfying QoS:

$\min_{i_n} Y_n(i_n) = w_t\left(T_n^{(c)} + T_n^{(w)}\right) + w_e\left(E_n^{(c)} + E_n^{(t)}\right) - w_u U_n(i_n)$ (14)

Subject to the constraints above ($\tau_1 \leq T_q$energy budget, resource limits), where w_t, w_e, w_u are weights and U_n is local utility (accuracy/throughput). Federated aggregation at the CS remains privacy-preserving: after each round, the CS updates the global parameter θ by weighted aggregation$\theta^{(t+1)} = \sum_{n\in N}\frac{D_n}{\sum_x D_x}\omega_n^{(t+1)}$ (15)

And optionally broadcasts g^{best} (best particle) or refined guidance back to F-APs to steer the next DRPSO iteration. This joint formulation integrates the FL timing and energy Equations ((2)–(5)), including computation and transmission timings with ripple-modulated swarm updates, enabling PSO particles to adapt resource allocation such that $\tau_1 \leq T_q$, while minimizing total energy consumption across the IoT-MEC network. The framework has three hierarchical levels - Edge Clusters and Devices, Edge Al, and Heterogeneous Federated Learning (HeteroFL), to facilitate collaborative learning and adaptive optimization in distributed IoT-MEC.

Edge Clusters & Devices Layer: It contains a variety of IoT devices (e.g., smart car parking, mobile healthcare, agriculture monitoring), each of which is a local learner. Sensors gather real-time data and run a local model at low resources, where each edge node x takes local updates to reduce its local loss function:

$\min_{\omega_x} F_x(\omega_x) = \frac{1}{D_x}\sum_{y=1}^{D_x} f(\omega_x; i_{xy}, j_{xy})$ (16)

Where D_x denotes the local dataset size, and $f\ (.)$ represents the loss over samples (i_{xy}, j_{xy}).

Edge AI Layer: The Edge AI servers (Edge Nodes) receive updates of local devices in their clusters. All the Edge Al nodes perform a local aggregation with swarm intelligence (DRPSO). The model parameters updates of velocity and position of the DRPSO are represented as follows:$v_x^{t+1} = \alpha v_x^t + c_1 r_1(p_x - \omega_x^t) + c_2 r_2(g - \omega_x^t) + \eta \sin(\lambda t)$ (17)

$\omega_x^{t+1} = \omega_x^t + v_x^{t+1}$ (18)

Here, the ripple term $\eta \sin(\lambda t)$ models the adaptive "wave-like" exploration to balance convergence and diversity during optimization.Each edge node computes its optimized local model ω_x^{t+1} and forwards it to the global computation server in the HeteroFL layer.

Heterogeneous Federated Learning (HeteroFL) Layer: The central coordinator aggregates the received local models from different Edge AIs into a global model using a weighted averaging mechanism:$\omega_{t+1}^{global} = \sum_{x=1}^{N}\frac{D_x}{D_{total}}\omega_x^{opt}$ (19)

The HeteroFL module accounts for heterogeneity in data, computation, and communication capacities of different devices. It ensures scalability and energy efficiency by dynamically adjusting learning rates, communication intervals, and model sizes.

Adaptive Resource Management Objective: The total federated learning time τ_1 and energy consumption E_{FL} are minimized jointly: $\min_{\{f_x, R_x\}} \tau_1 + \lambda E_{FL}$

subject to QoS and energy constraints, where f_x is the CPU frequency of device x, R_x is the transmission rate, and λ is the balancing coefficient.

4.4 Joint optimization

The joint optimization in DRAFSO tightly couples clustering, routing, and computation offloading into a single adaptive decision process that minimizes a combined energy-latency cost while respecting device, network, and MEC capacity constraints: each device x generates tasks with data size d_x and CPU cycles requirement C_x and chooses either local execution or offloading (binary decision $i_x \in \{0,1\}$ where $i_x = 1$ means offload). Local execution time is $T_x^{loc} = \frac{C_x}{f_x^{loc}}$and its energy (using a standard DVFS model) is $E_x^{loc} = k_x C_x (f_x^{loc})^2$, where f_x^{loc} is the device CPU frequency and k_x is the effective capacitance coefficient. Offloading incurs a transmission time $T_x^{ti} = \frac{d_x}{b_{x\to m}} + \tau_{x\to m}$ and transmission energy $E_x^{ti} = p_x^{ti} T_x^{ti}$ when bi-m sent to MEC node m (with bandwidth $b_{x\to m}$, propagation delay $\tau_{x\to m}$, and transmit power p_x^{ti}), plus remote execution time $T_{x,m}^{mec} = \frac{C_x}{f_m^{mec}}$ and MEC energy share $E_{x,m}^{mec}$. T

Therefore define per-task latency $L_x = (1 - i_x)T_x^{loc} + i_x\left(T_x^{ti} + T_{x,m}^{mec}\right)$ and per-task energy $E_x = (1 - i_x)E_x^{loc} + i_x\left(E_x^{ti} + E_{x,m}^{mec}\right)$.

The global objective is a weighted minimization min $\sum_x(\omega_E E_x + \omega_L L_x)$ (or equivalently a bi-objective optimization minimizing $\sum_x E_x$ and $\max x, L_x$), Subject to

(1) Device energy constraints $E_x^{consumed} \le E_x^{cap}$

(2) MEC CPU capacity $\sum_{x:i_x=1} C_x \le F_m^{cap}$ for each MEC m,

(3) Link bandwidth constraints$\sum_{x\in U_{x\to m}} d_x / T^{slot} \le b_{x\to m}$,

(4) Routing/flow-conservation constraints for each node (incoming flow = outgoing flow $\pm$ generated/consumed tasks)

(5) Clustering assignment constraints $\sum_c a_{x,c} = 1$ with binary assignment $a_{x,c} \in \{0,1\}$ and CH selection variables that enforce CH capacity and residual-energy-aware rotation. This formulation naturally produces MINLP because of binary offloading/assignment variables and nonlinear energy/latency relations; DRAFSO solves it approximately and distributive by using ripple-driven local updates (to adjust $a_{x,c}$ and routing preferences) and federated swarm intelligence (to update offloading probabilities and MEC resource allocations), thereby achieving near-optimal trade-offs between lifetime, energy consumption and response time while maintaining scalability and privacy

$Minimize\ Y = \sum_x(\omega_E E_x + \omega_L L_x)$ (20)

Subject to:

$E_x = (1 - i_x)k_x C_x (f_x^{loc})^2 + i_x\left(p_x^{ti} Y_x^{ti} + E_{x,m}^{mec}\right)$ (21)

$L_x = (1 - i_x)\frac{C_x}{f_x^{loc}} + i_x\left(\frac{d_x}{b_{x\to m}} + \tau_{x\to m} + \frac{C_x}{f_m^{mec}}\right)$ (22)

$$\sum_{x:Z_x=1 \wedge m(x)=m} C_x \le F_m^{cap} \ \forall m,$$

$$E_x^{consumed} \le E_x^{cap} \ \forall x,$$

$$\sum_{c} a_{x,c} = 1 \; a_{x,c} \in \{0,1\} \;\; \forall x \; (23)$$

$$flow\ conservation: \sum_{y \in N_{in}(v)} f_{yv} - \sum_{k \in N_{out}(v)} f_{vk} = s_v \;\; \forall v,$$

$$i_x \in \{0,1\}, \qquad a_{x,c} \in \{0,1\}$$

Device energy, MEC resource constraints, and communication bandwidth are all simultaneously considered in the MINLP optimization problem. Nodes dynamically adjust their offloading decisions, routing paths, and cluster associations through ripple-driven adaptive updates. Federated swarm intelligence ensures privacy preservation and decentralized efficiency.

4.5 Algorithm: Dynamic Ripple-driven Adaptive Federated Swarm Optimization

Input: $D = \{d_1, d_2, \dots, d_N\}$: Set of N IoT devices; $M = \{m_1, m_2, \dots, m_K\}$: Set of k MEC servers; Maximum iterations: T_{max}; Ripple propagation coefficient: α; Learning rate: η; Initial swarm population: I_x^0

Output: Optimal resource allocation R^*; Energy-efficient scheduling for IoT-MEC network

Step 1: Initialize Swarm Population

Step 1.1: Initialize positions I_x^0 and velocities V_x^0 of x-th particle (solution) randomly.

Step 1.2: Evaluate initial fitness for each particle: $F(I_x) = w_1.E(I_x) + w_2.T(I_x)$ (24)

Where: $E(I_x)$ = total energy consumption; $T(I_x)$ = latency of resource allocation; w_1,w_2= weighting factors

Step 2: Federated Learning Local Update (IoT Devices)

For each IoT device d_x : $\theta_x^{t+1} = \theta_x^t - \eta \nabla L_x(\theta_x^t)$ (25)

Where: θ_x^t= local model parameters at iteration; $L_x(\theta_x)$ = local loss function of device d_x; η = learning rate.

IoT devices upload local parameters θ_x^{t+1} to MEC servers (F-APs).

Step 3: Ripple-Driven Adaptive Swarm Update

For each particle x (solution):

Step 3.1: Compute ripple effect based on neighborhood best: $R_x^t = \alpha \sum_{y \in N_x} \frac{I_y^t - I_x^t}{\|I_y^t - I_x^t\| + \epsilon}$ (26)

Where: N_x = neighborhood of particle x; ϵ = small constant to avoid division by zero

Step 3.2: Update velocity and position:

$V_x^{t+1} = \omega V_x^t + c_1 r_1 (P_x^{best} - I_x^t) + c_2 r_2 (G^{best} - I_x^t) + R_x^t$ (27)

$I_x^{t+1} = I_x^t + V_x^{t+1}$ (28)

Where: ω = inertia weight; c_1, c_2 = cognitive and social coefficients; r_1, r_2 = random numbers; P_x^{best} = personal best position; G^{best} = global best position

Step 4: Global Aggregation (MEC Server / CS)

Step 4.1: Collect updated particle positions I_x^{t+1} from all local IoT devices.

Step 4.2: Compute aggregated global model for resource allocation: $\Theta^{t+1} = \frac{1}{N} \sum_{x=1}^{N} \theta_x^{t+1}$ (29)

Step 4.3: Broadcast updated global parameters back to IoT devices.

Step 5: Convergence Check

If $t \geq T_{max}$ or $\Delta F(G^{best}) < \epsilon$, stop
Else, go to Step 2

Step 6: Output Optimal Resource Allocation

Optimal energy-efficient allocation: $R^* = I_{G^{best}}$ (30)

Provides minimal energy consumption and reduced latency.

DRAFSO technique guarantees a scalable and energy-effective system management in IoT-MEC systems through the incorporation of FL with ripple-enhanced swarm optimization. IoT devices locally learn using private data, but only model parameters are shared with the MEC servers to be aggregated globally. Resource allocations are simulated as particles that change their position and velocity based on the global and personal bests, depending on the ripple effects that help to explore and avoid early convergence. Based on a fitness function of task delay and energy consumption, particles are optimally vested and brought to a final configuration (particles) to obtain adaptive, collaborative, and energy-efficient resource management in dynamic IoT-MEC setups.

5. Results and Discussions

Through multiple iterations, particles are optimized to reach a good configuration that not only minimizes latency and energy but also does not violate the ability to be flexible. The hybrid strategy is an effective tool to equally balance exploration, exploitation, and learning to manage the IoT-MEC resources optimally in the dynamic network setting. Multiple candidate solutions are used to initiate the swarm population,and the movements of particles are developed through ripple-driven adaptation to effectively search the search space. FL provides privacy-preserving collaborative enhancement, where IoT devices update local models that are combined by MEC servers into a global model. The simulations are run on different network sizes and iterations and trained on the MATLAB R2023a algorithm, Python 3.11 with Tensorflow and PyTorch federated learning, and Network simulation with NS-3. The hardware configuration consists of an Intel Core i7 processor, NVIDIA graphics card, 32GB RAM and connected nodes that serve as IoT devices and MEC servers.

Table 3. Hyper-parameter Settings

Hyper parameter	**Symbol**	**Value/Range**
Swarm population size	N_p	30-50
Maximum iterations	T_{max}	100
Inertia weight	ω	0.5-0.9
Cognitive coefficient	c_1	1.5
Social coefficient	c_2	1.5
Ripple propagation coefficient	α	0.2-0.5
Learning rate (IoT devices	η	0.01
Small constant for stability	ϵ	10^{-6}
Task size range	-	1-10 MB
Network bandwidth	-	100-500 Mbps

Table 3 illustrates how the DRAFSO framework effectively optimizes resource allocation by utilizing key hyperparameters. Exploration and exploitation are controlled by swarm size and iteration count, while local and global search are balanced by the inertia weight. The cognitive and social components direct particles toward their optimal positions on a global and individual scale, while the ripple coefficient facilitates exploration. A small constant ensures computational stability, and IoT devices use a defined learning rate to update local models. The framework's performance can be assessed in terms of resource usage, task completion delay, and energy efficiency by modeling realistic IoT-MEC scenarios with different task sizes and network bandwidth.

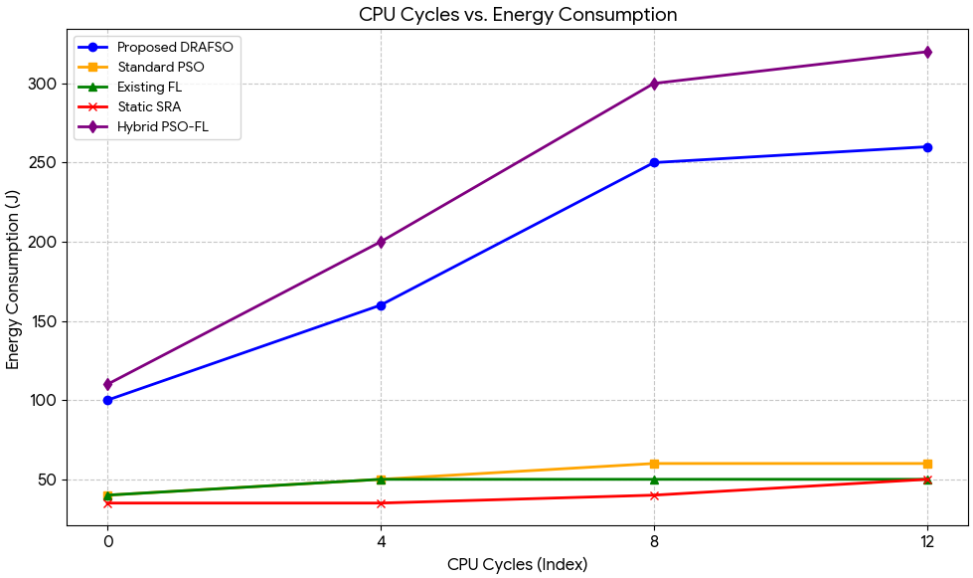


Fig. 5 CPU cycles based on IoT device Vs, Energy consumption

When CPU cycles and energy consumption are compared in Fig. 5, it is evident that the proposed DRAFSO model outperforms PSO, FL, SRA, and Hybrid PSO-FL in terms of computational efficiency and energy consumption. DRAFSO maintains optimal energy utilization as CPU cycles rise, successfully balancing processing efficiency and demand. Energy consumption changes with data size are shown in Fig.6 where DRAFSO progressively increases from 0 to 15 as data increases from 0 to 12 units. DRAFSO is perfect for dependable and sustainable IoT operations because of its exceptional scalability and energy efficiency are demonstrated by its consistent performance. DRAFSO efficiently lowers energy usage while preserving high performance under various workloads.

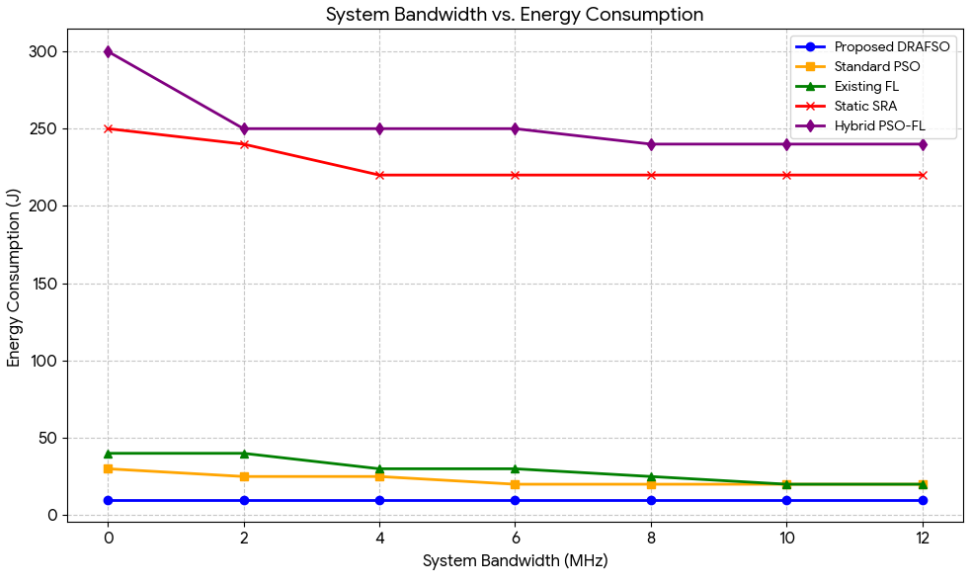


Fig. 6 Data size vs. Energy consumption

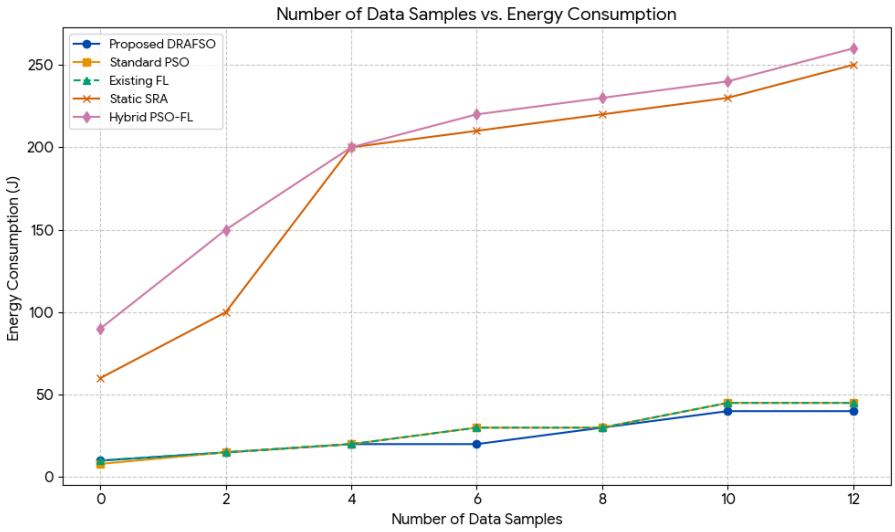


Fig. 7 Number of data samples vs. Energy consumption.

The relationship between energy consumption and the number of data samples for different optimization models is shown in Fig.7. Superior efficiency and stability are demonstrated by the proposed DRAFSO model, which gradually increases energy consumption from 10 to 40 as the number of data samples increases from 0 to 12. These findings demonstrate that DRAFSO efficiently reduces energy usage, leading to improved scalability and performance for processing large amounts of IoT data.

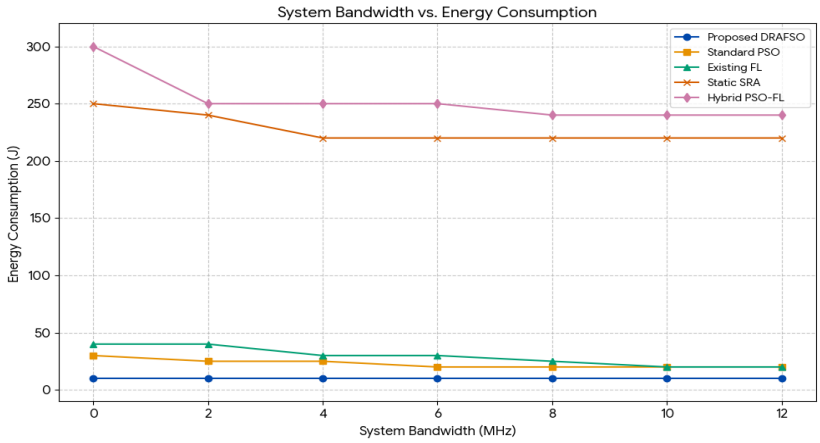


Fig. 8 System Bandwidth vs. Energy consumption

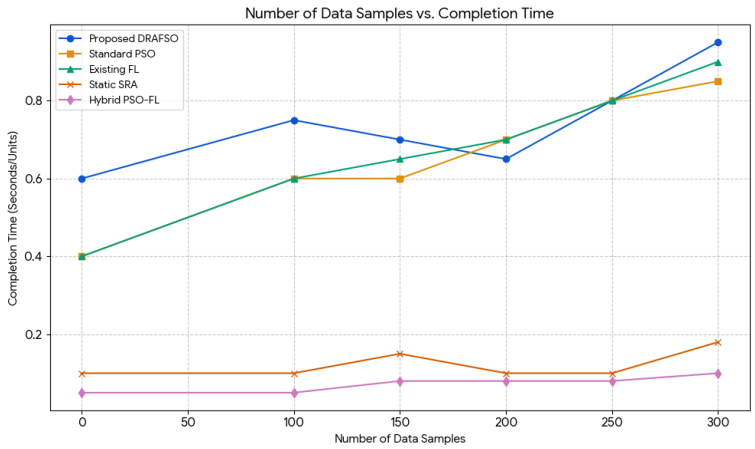


Fig. 9 Number of data samples vs. Completion time

Fig.8 illustrates how system bandwidth affects energy usage for various optimization strategies. The proposed DRAFSO model exhibits high efficiency by maintaining a consistent low energy consumption of 10 across all bandwidth levels. The superior energy efficiency and stability of DRAFSO in IoT systems is demonstrated by the higher and fluctuating energy usage of PSO, FL, SRA, and Hybrid PSO-FL. The relationship between task completion time and the number of data samples across various optimization techniques is shown in Fig. 9. As samples increase from 0 to 300, the proposed DRAFSO exhibits progressively shorter completion times, rising from 0.6 to 0.95. The efficiency of DRAFSO in managing increasing workloads is demonstrated by the higher and less consistent completion times of PSO, FL, SRA, and Hybrid PSO-FL in comparison.

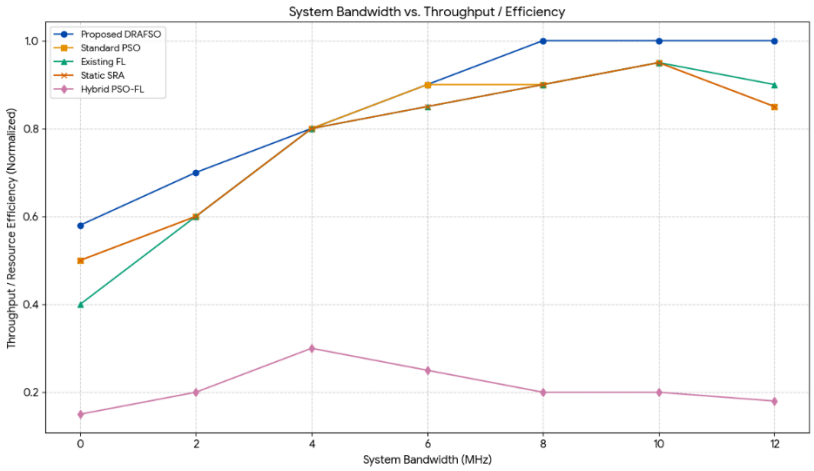


Fig. 10 Number of IoT devices vs. System bandwidth

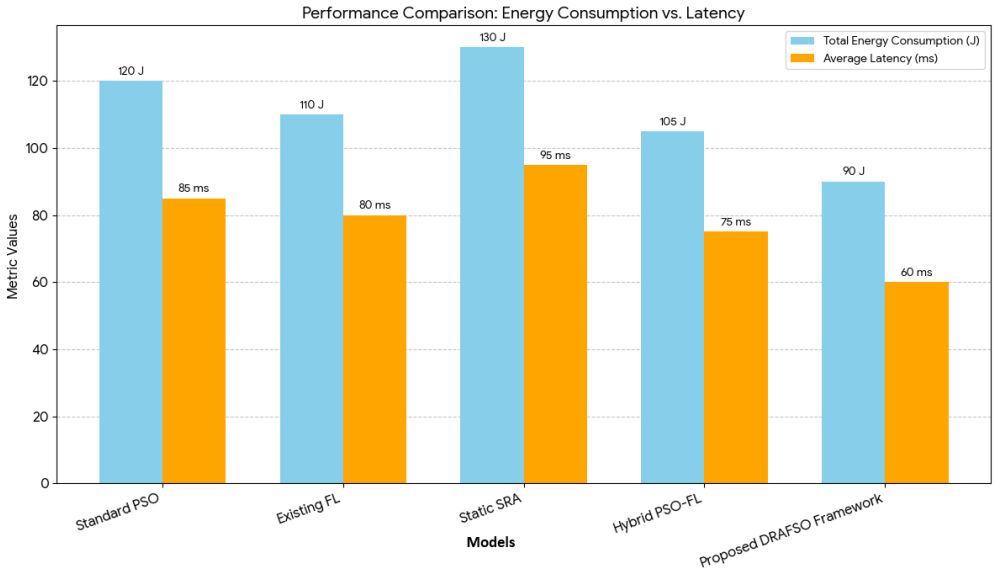


Fig. 11 Comparison of Total energy consumption and average latency of proposed and existing systems

The proposed DRAFSO continuously beats the standard PSO, existing FL, SRA, and Hybrid PSO-FL across all iterations, as the performance comparison illustrates in Fig. 10. The superior resource optimization and adaptability of DRAFSO in dynamic IoT-MEC environments are highlighted by the slower convergence and lower peak performance of other approaches.Fig.11 compares the proposed DRAFSO framework with PSO, FL, SRA, and hybrid PSO-FL in terms of energy consumption and latency. Traditional methods exhibit higher energy use and latency due to static or non-adaptive designs. While FL and hybrid PSO-FL improve efficiency, DRAFSO achieves the lowest energy consumption and latency through ripple-driven adaptive updates and federated swarm intelligence, enabling balanced resource allocation, faster convergence, and enhanced performance across heterogeneous IoT-MEC networks.

Table 4. Comparison of performance measures (Task Completion Rate, Convergence Speed, Resource Utilization, Communication Cost and Scalability)

Method	**Task Completion Rate (%)**	**Convergence Speed**	**Resource Utilization (%)**	**Communication Cost**	**Scalability**
Standard PSO	85	Medium	70	High	Low
Federated Learning (FL)	88	Medium-High	75	Medium	Medium
Static Resource Allocation (SRA)	80	Low	65	High	Low
Hybrid PSO-FL	90	High	78	Medium-Low	Medium-High
Proposed DRAFSO Framework	95	Very High	85	Low	High

Table 4 will compare the key performance indicators of various resource management practices. The proposed DRAFSO framework has the best results compared with Standard PSO, FL, SRA, and Hybrid PSO-FL, as it has the highest task completion rate and resource utilization, quickest convergence, lowest cost in communication, and the best scalability. This shows the efficiency and flexibility of DRAFSO in the networks of IoT-MEC.Fig.12 analyzes the performance of the classification between the other methods. The proposed DRAFSO model has the best accuracy, precision, recall, and F1-score as compared to Standard PSO, FL, SRA, and Hybrid PSO-FL. Such findings indicate a higher predictability of DRAFSO together with its balanced performance and general effectiveness in managing resources and task allocation in the IoT-MEC networks.

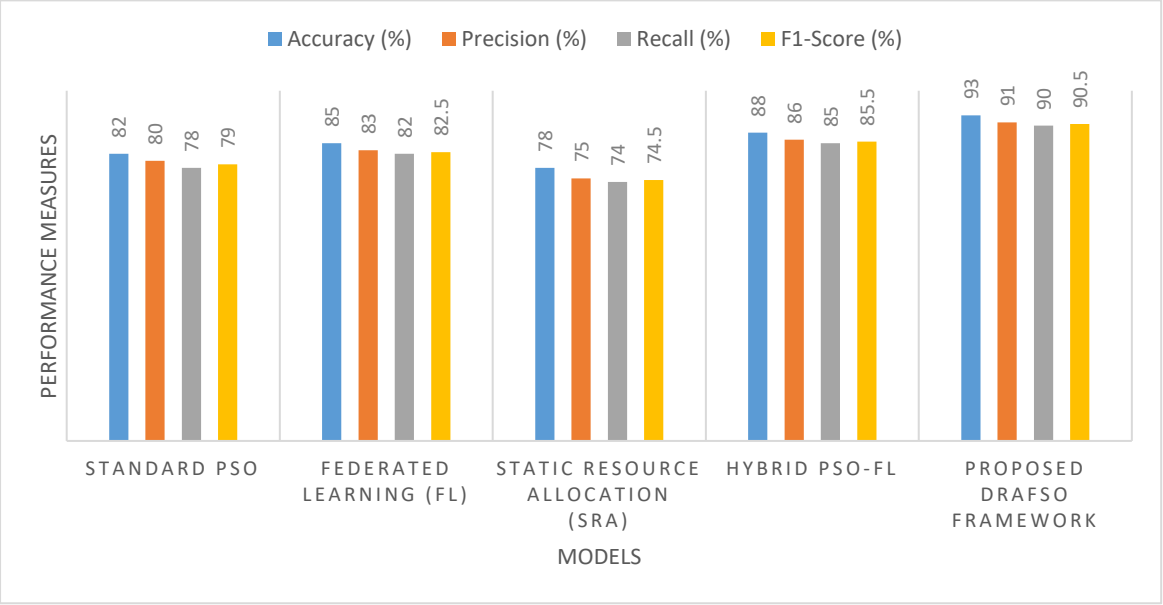


Fig. 12 Comparison of performance measures (Accuracy, Precision, Recall and F1 score)

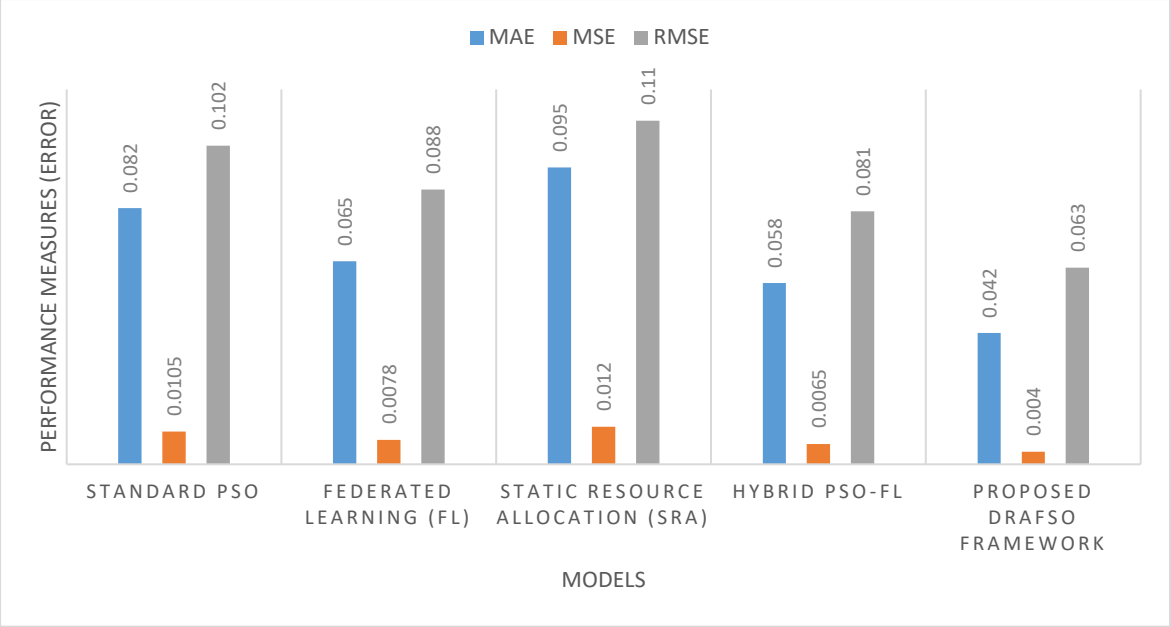


Fig. 13 Comparison of performance measures (Error)

Fig.13 makes comparison of various methods based on MAE, MSE and RMSE as errors that are used to predict. The proposed framework of DRAFSO obtains the lowest results in all measures, which beats the Standard PSO, FL, SRA, and Hybrid PSO-FL. This means that prediction accuracy and reliability are increased in IoT-MEC networks and that management of resources is more accurate. The accuracy of the proposed DRAFSO system in training, validating, and testing during epochs are represented in Fig.14. The accuracy increases with the number of epochs and validation and testing are directly below the performance on training. This shows the stable convergence, practical learning and robust generalization of the model, which explains its reliability and robustness in managing resources in IoT-MEC networks.

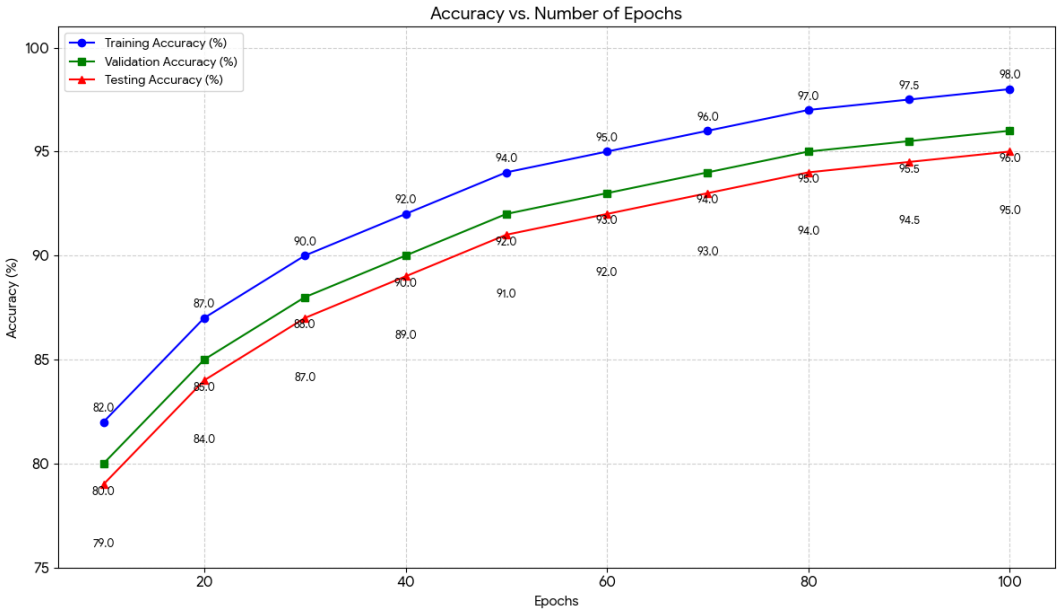


Fig. 14 Comparison of Training, Validation, and Testing Accuracy vs. epochs of the proposed system

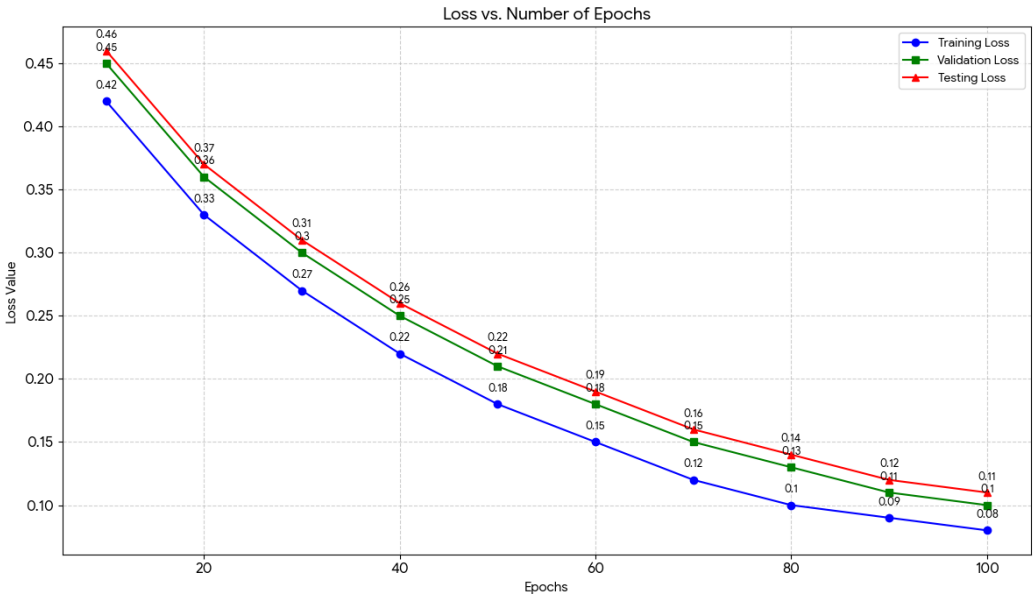


Fig. 15 Comparison of Training, Validation and Testing Loss vs. epochs of the proposed system

Fig.15 presents the training, validation, and testing loss of an offered DRAFSO system on epochs. The loss reduces progressively with the increase in epochs and violence and testing come next respectively. This suggests good learning, convergent stability, and good generalization, which proves the reliability and robustness of the model in resource management of IoT-MEC networks.

6. Conclusions

This work presents a Dynamic Ripple-Driven Adaptive Federated Swarm Optimization (DRAFSO) framework for scalable and energy-efficient resource management in IoT-MEC networks. By combining ripple-inspired adaptive updates with federated swarm intelligence, DRAFSO enables decentralized optimization, collaborative decision-making, and privacy preservation across heterogeneous MEC nodes. The framework jointly optimizes clustering, routing, and computation offloading, ensuring balanced resource allocation and minimal latency. Experimental evaluations demonstrate the superiority of DRAFSO over existing methods, achieving a 95% task completion rate, fastest convergence, 85% resource utilization, lowest communication cost, and high scalability. In predictive tasks, the framework attains 93% accuracy, 91% precision, 90% recall, and 90.5% F1-score, with minimal errors (MAE = 0.042, MSE = 0.004, RMSE = 0.063). Training, validation, and testing analyses show steady accuracy improvement and decreasing loss, indicating stable convergence and strong generalization. Overall, DRAFSO provides a robust, energy-aware, and adaptive solution, enhancing system lifetime, reducing energy consumption, and improving network performance. This framework establishes a practical and scalable approach for next-generation IoT-MEC resource management, supporting real-time adaptability in dynamic network environments.

Author contributions

Conceptualization of the study,Methodology, Data interpretation and Manuscript was done by M.Velvizhi, Supervision is done by D.Saraswady

Funding

No funding

Data Availability

The datasets generated and/or analysed during the current study are available in the

https://www.kaggle.com/datasets/ziya07/multi-tier-iot-resource-allocation-dataset

Declarations

Consent for publication

Not Applicable.

Ethics approval and consent to participate

Not Applicable

Competing interests

The authors declare no competing Interests.

References

1. Rahmani, A. M., Haider, A., Moghaddasi, K., Gharehchopogh, F. S., Aurangzeb, K., Liu, Z., & Hosseinzadeh, M. (2025). Self-learning adaptive power management scheme for energy-efficient IoT-MEC systems using soft actor-critic algorithm. *Internet of Things*, *31*, 101587.

2. Khan, S., Ali-Pour, A., Avgeris, M., Gascon-Samson, J., &Leivadeas, A. (2025). FEDORA: Federated Ensemble Reinforcement Learning for DAG-based task offloading and resource allocation in MEC. *IEEE Internet of Things Journal*.

3. Ahmed, M., Khan, W. U., Alrayes, F. S., Said, Y., Al-Sharafi, A. M., Kim, M. H., & Ullah, I. (2025). Joint Encryption and Optimization for 6G MEC-Enabled IoT Networks. *IEEE Access*.

4. Jiang, W., Xiao, Z., Qian, L. P., Qin, S., Feng, G., & Wu, Y. (2025). Joint Communication and Computing Resource Allocation for Energy Efficient Hierarchical Federated Learning in Marine Internet of Things. *IEEE Transactions on Network Science and Engineering*.

5. Mrabet, M., &Sliti, M. (2025). Towards Secure, Trustworthy and Sustainable Edge Computing for Smart Cities: Innovative Strategies and Future Prospects. *IEEE Access*.

6. Ros, S., Ryoo, I., & Kim, S. (2025). DRL-Driven Intelligent SFC Deployment in MEC Workload for Dynamic IoT Networks. *Sensors*, *25*(14), 4257.

7. Chen, J., Mi, J., Guo, C., Fu, Q., Tang, W., Luo, W., & Zhu, Q. (2025). Research on Offloading and Resource Allocation for MEC with Energy Harvesting Based on Deep Reinforcement Learning. *Electronics*, *14*(10), 1911.

8. Ismail, A. A., Khalifa, N. E., & El-Khoribi, R. A. (2025). A survey on resource scheduling approaches in multi-access edge computing environment: a deep reinforcement learning study. *Cluster Computing*, *28*(3), 184.

9. Tan, R., Li, R., Yu, X., Chen, X., Xu, X., & Zhao, Z. (2025). Pareto Actor-Critic for Communication and Computation Co-Optimization in Non-Cooperative Federated Learning Services. *IEEE Transactions on Mobile Computing*.

10. Zakaryia, S. A., Mead, M. A., Nabil, T., & Hussein, M. K. (2025). Task offloading for multi-UAV asset edge computing with deep reinforcement learning. *Cluster Computing*, *28*(7), 462.

11. Wang, Y., Su, S., & Wang, Y. (2025). Attention-augmented multi-agent collaboration for Smart Industrial Internet of Things task offloading. *Internet of Things*, *31*, 101572.

12. Mohamed, H., Marouane, H., & Fakhfakh, A. (2025). A Systematic Review Of Multi-Agent Systems And Mobile Edge Computing In Intelligent Transportation Systems. *Journal of Soft Computing and Data Mining*, *6*(1), 169-181.

13. Kumar, A., Gahlawat, R., Thakur, A., & Pahuja, D. (2025). A Hybrid Deep Learning Framework for IoT Network Intrusion Detection System.

14. Rinesh, S., & Pandian, A. (2025). Ultra-Low Latency Optimization in 6G Consumer Electronics Using Synergistic Graph-Reinforcement Learning for Efficient Communication. *IEEE Transactions on Consumer Electronics*.

15. Hu, Y., Chen, M., Sun, H., Wang, Y., & Cang, Y. (2025). Joint Optimization of Resource Allocation and Task Offloading Strategies in Multi-Cell Dynamic MEC Systems Using Multi-Agent DRL. *IEEE Access*.

16. Zheng, T. X., Wang, X., Chen, X., Mao, D., Shi, J., Pan, C., ... & Li, Z. (2025). Reconfigurable Intelligent Surface-Enabled Green and Secure Offloading for Mobile Edge Computing Networks. *IEEE Internet of Things Journal*, (99), 1-1.

17. Ali, E. M., Abawajy, J., Lemma, F., & Baho, S. A. (2025). Analysis of Deep Reinforcement Learning Algorithms for Task Offloading and Resource Allocation in Fog Computing Environments. *Sensors*, *25*(17), 5286.

18. Kumar, S. S., & Kannan, B. (2025, August). Enhancing Industrial IoT Security with AI and Cloud Computing: A Review of Threats and Solutions. In *2025 International Conference on Modern Sustainable Systems (CMSS)* (pp. 1082-1089). IEEE.

19. Barth, B., Baradie, S., Keuker, C., Zaitsev, R., Guruvayoorappan, G., & De Cola, T. (2025). MEC in 3D-Networks: Supporting Application Controlled QoS. *IEEE Access*.

20. Mrabet, M., &Sliti, M. (2025). Towards Secure, Trustworthy and Sustainable Edge Computing for Smart Cities: Innovative Strategies and Future Prospects. *IEEE Access*.

21. Guo, R., Zhou, L., Li, L., Song, Y., & Xie, X. (2025). Dependent Task Graph Offloading Model Based on Deep Reinforcement Learning in Mobile Edge Computing. *Electronics*, *14*(16), 3184.

22. Yang, L. (2025). Heterogeneous Internet of Things Big Data Analysis System Based on Mobile Edge Computing. *Journal of Information & Knowledge Management*, *24*(03), 2450047.