International Academic Institute for Science and Technology

International Academic Journal of Science and Engineering
Vol. 12, No. 3, 2025, pp. 267-282.

ISSN 2454-3896

International Academic Journal of Science and Engineering

www.iaiest.com

Hierarchical Multi-Leader Multi-Follower Stackelberg Game for Dynamic Computation Offloading in MEC-Enabled IoT Systems

Shreenivas Badada [a], Dr. Akkalakshmi Muddana [b]

[a] Research Scholar, Department of Computer Science and Engineering, GITAM (Deemed to be University), Hyderabad, Telangana, India. E-mail: shreenivasbadada@gmail.com, Orcid: https://orcid.org/0009-0005-6728-9536

[b] Professor, Department of Computer Science and Engineering, GITAM (Deemed to be University), Hyderabad, Telangana, Hyderabad, India. E-mail: amuddana@gitam.edu, Orcid: https://orcid.org/0000-0002-9721-6507

Abstract

Mobile Edge Computing (MEC) has become one of the enabling factors in the delivery of latency-sensitive, energy-constrained Internet of Things (IoT) applications. Nevertheless, offloading computation in heterogeneous and dynamic MEC-IoT settings is still a challenge due to resource contention, dynamic workload fluctuations, and economic issues between service providers and users. This paper proposes a solution to these problems by introducing an Enhanced Hierarchical Multi-Leader Multi-Follower (MLMF) Stackelberg game model for distributed computation offloading. According to the proposed model, MEC servers are leaders that use an Iterative Stackelberg Pricing Algorithm (ISPA). In contrast, IoT devices are followers that use an Iterative Proximal Offloading Algorithm (IPOA) to select the optimal task allocation. The interaction between the IPOA and ISPA will ensure that a capacity-constrained Stackelberg equilibrium is reached. Massive simulations in MATLAB 2023b demonstrate that the proposed framework achieves 60-97% reduction in execution delay compared to baseline schemes, 30-35% energy savings with effective offloading, a fairness index of 0.95-0.97 under Jain's fairness index, and utility optimization of the system through balanced economic incentives. The ablation experiment also confirms that either of the two ablation conditions can ruin performance when either IPOA or ISPA is removed, indicating that hierarchical joint optimization is required. The proposed MLMF system offers a large-scale, cost-effective, and justice-oriented solution to the MEC-enabled IoT systems.

Keywords: Mobile Edge Computing, Joint Iterative Proximal Offloading Algorithm, Multi-Leader Multi-Follower, Distributed Optimization, Task Offloading Strategy, Load Balancing, Edge Intelligence Systems.

DOI: 10.71086/IAJSE/V12I3/IAJSE1264
Received: 09 July 2025/Revised: 14 August 2025/Accepted: 09 September 2025/Published: 30 September 2025

Introduction

Recently, MEC has been identified as a paradigm shift in next-generation networks, which allows the provision of low-latency computation through the deployment of servers as close to end users and IoT devices as possible. As latency-sensitive, resource-intensive IoT applications (augmented reality, autonomous vehicles, and industrial automation) continue to grow exponentially, the need for efficient and dynamic computation offloading policies has become increasingly urgent. The key concept of MEC is that tasks can be distributed to edge servers deployed at the wireless network edge. Wireless devices can solve the problem of a lack of computational resources, reduce the delay of computing services (Lin et al., 2020; Fan et al., 2021), and save energy (Wang et al., 2021; Zhang et al., 2021). Conventional offloading methods are used under high-static conditions; however, they are no longer applicable to highly dynamic and multi-user MEC systems, where resource contention and network heterogeneity play important roles in performance. Energy harvesting (EH) is a technology that can overcome these limitations. It can collect ambient recyclable energy, such as solar radiation, wind, and human motion energy (Sudevalayam & Kulkarni, 2010), and therefore enables self-sustenance and continued functioning (Ulukus et al., 2015). To solve this, game-theoretic models, and in particular the Stackelberg model, have increasingly been used to model hierarchical terms in which MEC service providers (leaders) and users (followers) interact under strategic conditions. The MEC server is seen as a leader that, in a Stackelberg game, informs the IoT users about its service price or resource allocation policy first, and the latter as rational followers who then select their offloading strategy to reduce the task execution delay and energy consumption (Chen et al., 2020). This hierarchy is practical in reflecting the real-world MEC systems with service providers possessing price control and users incentivized to make cost-efficient decisions. Stackelberg enables distributed optimization and supports the differentiation of Quality of Service (QoS) needs, dynamic network constraints, and energy limits, and thus such a model can be scaled to support large-scale IoT systems. Recent studies have extended the Stackelberg game to dynamic multi-user environments to utilize variations over time as well as user mobility. These improved models allow servers to dynamically modify prices or resource allocations across several time steps while considering user behavior and task urgency. For example, (Wang et al., 2017) introduced a multi-leader Stackelberg game to plan computation offloading on dense MEC networks, which provides a superior level of fairness and energy efficiency between conflicting IoT devices. Within such frameworks, users actively analyze their local resources, transmission rates, and energy budgets to determine whether they should process tasks locally or hand them over to edge servers. In addition, in combination with reinforcement learning, Stackelberg-based methods are further strengthened in unknown or quickly evolving environments, where the best strategies are learned via interaction and feedback (Ren et al., 2025). Moreover, the framework of the Stackelberg game is particularly advantageous to MEC-based IoT systems because it can balance economic incentives with optimization at the system level.

The service providers want to maximize revenue, and the users want to minimize their operational costs; thus, a Stackelberg equilibrium results in neither party being interested in unilateral deviation. Game-theoretic equilibrium is vital for the sustainable provision of MEC services in energy-limited IoT systems. Increasingly complex IoT ecosystems (with multiple MEC servers and heterogeneous devices) are being modelled using Stackelberg-based models that are extended to multi-server and multi-tier models (Gong et al., 2024). Overall, the Stackelberg game is a versatile and robust tool that can be used to compute offloading dynamics in MEC-IoT settings. It provides a mathematically based solution to distributed resource management, motivation to collaborate, and assurance of QoS guarantees; thus, it is a foundational block in the design of future edge intelligence systems. The three significant contributions are

- This study presents a enhanced MLMF Stackelberg Game-based framework in MEC-enabled IoT systems, where the MEC servers serve as leaders and the IoT components serve as followers attempting to optimize their offloading choices. This hierarchical model is highly effective in describing the interactions between heterogeneous devices and servers with dynamic workloads.
- The framework combines an IPOA of follower-side task allocation with an ISPA of leader-side resource pricing. Initially, they all execute the plan to ensure convergence to a Stackelberg equilibrium, which ensures that the loads are evenly distributed, fairness, and that the system utility is optimized.
- A large-scale MATLAB-based simulation was performed with different numbers of devices, workloads, and server capacities to compare the proposed approaches to greedy offloading (GOA), centralized optimization (COA), and random offloading (ROA).

The rest of this paper will be organized as follows. Section 2 reviews related work on MEC-enabled IoT computation offloading, outlining available game-theoretic, centralized, and learning-based methods and identifying significant

research gaps. Section 3 introduces the proposed MLMF Stackelberg game model, comprising the system model, the leader-follower optimization formulations, the equilibrium analysis, and the joint IPOA-ISPA algorithm. Section 4 outlines the simulation configuration and performance measurement, provides a detailed comparison of the simulation's performance with baseline algorithms for delay, energy consumption, server load, fairness index, and system utility, and presents an ablation study. Lastly, Section 5 presents the conclusion of this paper and outlines possible future directions for research to scale the practice, adaptability, and real-world application of MEC-IoT systems.

Related Works

Studies on MEC-enabled IoT are mostly related to computation offloading to prevent latency and energy consumption in dynamic network conditions. The available literature addresses the issues of energy-efficient offloading, cooperative multi-server scheduling, task prediction (AI), and decision frameworks through reinforcement learning. Edge server/user hierarchical interactions have adopted game-theoretic models, particularly Stackelberg games, to model hierarchical interactions. Nevertheless, most methods target single-leader cases or do not consider fairness and scalability in large and diverse MEC-IoT systems.

In (Rong, 2023), the authors discussed approaches to make mobile edge computing (MEC) environments more energy efficient. As mobile devices become increasingly dependent on MEC to handle computation-intensive activities, energy consumption becomes a paramount concern. The article explains that the decision related to computation offloading should balance energy savings with latency needs. It proposes smart offloading systems that use sophisticated algorithms, including deep learning, Lyapunov optimization, and game theory to dynamically allocate resources and achieve context-aware offloading. They noted that in MEC offloading, energy optimization is a complex issue because it must be optimized holistically, considering the wireless channel characteristics, cellular mobility, and server load. They also described the complexity of making real-time decisions in extremely dynamic MEC networks and provided future research directions, such as AI-based adaptive algorithms and cross-layer optimization. Overall, this article offers useful information on how to design sustainable and high-performance MEC systems that optimize energy consumption without deteriorating the user experience.

In (Guo et al., 2024), the authors introduced new optimization models for MEC-based task offloading and execution scheduling. This study strives to enhance the quality of delay-assured services in scenarios where multiple MEC servers collaborate to fulfil user tasks. The authors suggest a combined optimization approach to combine offloading decisions with parallel task scheduling to decrease the total latency and enhance resource utilization. One of these is the development of a delay-conscious algorithm that distributes tasks among multiple servers while still attending to strict delay requirements. Simulations confirm the use of this approach, which greatly enhances service delay reduction and system throughput. The proposed model uses cooperative MEC and parallelism to enable more responsive and scalable edge computing systems. The article notes that there is an increasing demand for intelligent orchestration in applications with latency constraints, such as IoT applications and real-time analytics.

In (Zeng et al., 2025), the authors proposed a developed offloading architecture in which generative AI is incorporated to enable AI-generated content (AIGC) services in the Metaverse environment on an IoT network. The authors addressed the challenge of processing multimodal data, such as text, images, and 3D models, by introducing a multimodal parallel offloading scheme. This approach employs generative AI models to predict task characteristics and efficiently divide and distribute computation tasks across multiple edge servers. The suggested solution allows us to produce and interact in real time, which is the key to making the metaverse immersive. The structure assists in making smart decisions that are balanced in terms of the computing delay, network load, and energy efficiency. Massive simulations have proven this solution, demonstrating a significant decrease in latency and an increase in the service reliability. This research is an early step toward using AI to coordinate sophisticated offloading in future IoT and Metaverse applications to provide smooth, low-latency experiences in high-dynamism environments.

In (Dong et al., 2025), the authors introduced a smart offloading scheme aimed at delay-sensitive applications in MEC. To reduce service delay and network congestion, the authors suggest a martingale-based deep reinforcement learning (M-DRL) framework that optimizes service offloading and edge caching decisions. The M-DRL model incorporates martingale theory to achieve stable convergence of learning and improved adaptation under dynamic network settings. The algorithm can learn the best policies to offload tasks, considering their urgency, caching availability, and wireless resource constraints, to utilize the available content in the caches to minimize redundant transmissions. The simulation results indicate that the proposed approach is more effective than the current methods in terms of delay reduction, task success rate, and system efficiency. Their work is highly relevant to the creation of intelligent, adaptive MEC systems,

especially for real-time applications, such as AR/VR, autonomous driving, and industrial IoT, where latency is a key factor.

In (Li et al., 2024), the authors presented a new offloading framework that works with heterogeneous air-ground edge computing systems based on Non-Orthogonal Multiple Access (NOMA) technology. This study addresses the issue of dynamic computation task management of IoT devices in cases where Unmanned Aerial Vehicles (UAVs) and terrestrial edge servers collaborate. The authors developed a dynamic and power-efficient offloading approach that optimizes the task and power distributions and considers both time-varying wireless channel behaviors and user mobility. The model can be used to leverage the benefits of NOMA to allow more users to share the same spectrum resource more effectively, thereby enhancing spectral efficiency and minimizing energy usage in the system. The iterative algorithm is a low-complexity algorithm that solves optimization problems. The results of the simulation prove that the proposed approach is much better than the traditional schemes in terms of energy efficiency and delay performance. This work contributes to the implementation of integrated air-ground MEC in IoT-intensive and latency-sensitive applications.

According to the literature review, there are several crucial research gaps. The available literature on MEC offloading research mainly dwells on either single-leader Stackelberg, centralized optimization, or learning-based schemes, which are subject to scalability problems, high computational costs, and slow convergence in high-dynamics IoT settings. Although cooperative MEC and reinforcement learning are effective at reducing delays, they generally require a large amount of training and centralized coordination, which is not suitable for real-time large-scale deployments. In addition, the majority of the works fail to simultaneously maximize latency, energy utilization, fairness, and economic incentives with uneven device and server characteristics (Zhou et al., 2021; Zhang et al., 2024; Tu et al., 2022; Lim et al., 2020; Lim et al., 2021). The lack of interaction modeling of MLMF also means that they cannot be used in real-world MEC ecosystems to the extent that many edge providers interact with each other. These constraints are the direct inspiration of the proposed MLMF Stackelberg model, according to which a number of MEC servers are perceived as leaders and IoT devices as followers, allowing distributed decision-making, reaching a stable Stackelberg equilibrium, and balancing technical performance and economic efficiency. Therefore, the suggested IPOA-ISPA-based MLMF framework clearly counters the failure of previous studies by presenting scale, fairness, and flexibility in dynamically changing IoT workloads.

Materials and Methods

The proposed MLMF Stackelberg Game for MEC/IoT computation offloading is a two-step iterative process between the MEC server and IoT devices as illustrated in fig. 1. First, every MEC server, as a leader, charges a price per CPU cycle depending on its available computing power and expected demand. To address this, IoT devices, acting as followers, decide on the optimal portion of their computation tasks to offload to specific MEC servers by minimizing their respective cost functions, which are a function of server prices and task characteristics. The execution of this decision-making is carried out with the help of an IPOA. Simultaneously, servers optimize their resource loads using the ISPA, which uses iterative price adjustments.

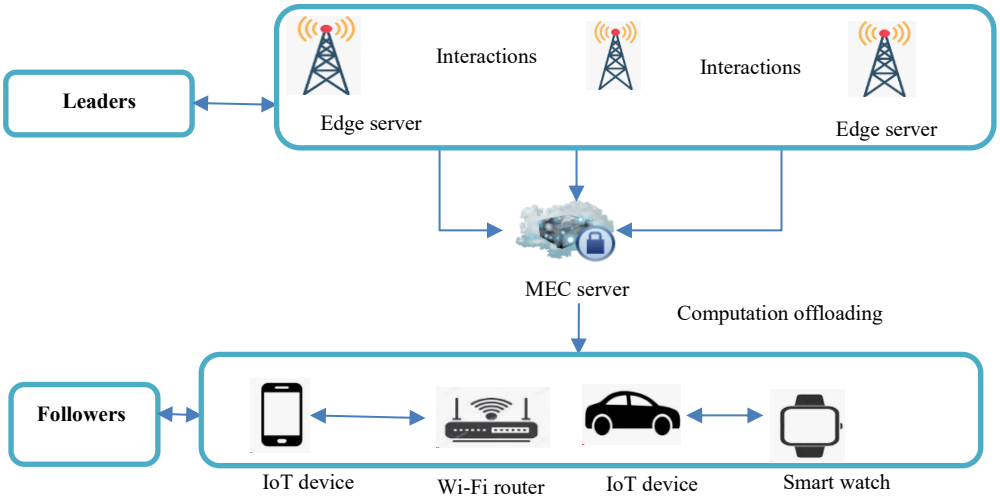


Fig. 1: System Model

This continues until the system attains a Stackelberg equilibrium, where no agent (server or device) is motivated to unilaterally alter its strategy. The hierarchy and distributed nature of this optimization guarantee dynamic resource balancing, energy saving, and minimization of delays in real-time MEC-IoT settings. Fig. 2 describes the proposed MLMF Stackelberg game model of MEC-based Internet of Things computation offloading. On the first layer, MEC servers (leaders) use the ISPA to compute dynamic resource pricing based on demand and capacity. Such pricing signals affect the IoT devices (followers) that use the IPOA to assign tasks between edge and local processing optimally. Bidirectional pricing and offloading interactions proceed iteratively until a Stackelberg equilibrium is reached. The architecture eventually guarantees reduced cost, energy conservation, equal resource distribution, and utility in heterogeneous IoT environments.

Stage I: Mec Server (Leader) Optimization

At the hierarchical MLMF Stackelberg game of MEC, every MEC server is a leader, that is, a player that attempts to maximize its profit by establishing optimal pricing measures. We use $S = \{1, 2,..., S\}$ to represent the set of MEC servers and $U = \{1, 2,..., U\}$ to represent the set of IoT users. MEC server $s \in S$ will set a price per CPU cycle, p_s, to maximize its own revenue in R. The generated revenue would be based on the overall quantity of CPU cycles offloaded to servers by all users. The user decision variables are the offloading ratios of the users, α_{us}, that represent the fraction of computation tasks that user $u \in U$ offloads to server s. Server revenue is provided by:

$$R_s p_s \sum_{u \in U} \alpha_{us} D_u C_u \tag{1}$$

Subject to the computational capacity constraint:

$$\sum_{u \in U} \alpha_{us} D_u C_u \leq F_s \tag{2}$$

The size of the task of user u (bits) is D_u, the required number of CPU cycles per bit is C_u, and F_s is the overall compute capacity of server s. The leader needs to predict as well the optimal-response behavior of the followers (IoT devices), which resolves their optimization in the announced prices $\{p_s\}$. The leader maximization thus emerges as a bi-level programming in which the leader maximizes p_s and expects the followers to respond:

$$\max_{p_s} R_s(p_s, \alpha^*_{us}(p_s)) \;\; s.t. \sum_{u} \alpha^*_{us}(p_s) D_u C_u \leq F_s \tag{3}$$

This encapsulates the Stackelberg structure, where MEC servers lead and users follow.

Stage II: Iot Device (Follower) Optimization

With the IoT devices operating as followers, the second phase of the Hierarchical multi-leader multi-follower Stackelberg Game is to minimize their respective computation costs and response delays to prices specified by the MEC servers (leaders). Each user $u \in U = \{1,2,...,U\}$ needs to determine the share $\alpha_{us} \in [0,1]$ of its compute task to be offloaded to MEC server $s \in S$, so that the sum of offloading and execution remains equal to one. The objective of the user is to reduce its total cost, which comprises the (i) communication delay (uplink transmission), (ii) execution delay (remote and local), and (iii) monetary cost incurred because of server pricing. The task size (in bits) is D_u, the CPU cycles per bit is C_u, the uplink rate between user u and server s is R_{us}, the local CPU frequency of user u is f_u, the CPU frequency of server s is f_s, and the price per CPU cycle per server s is p_s. The user cost function of user u is:

$$\min_{\alpha_{us}} J_u = \underbrace{\left(1 - \sum_{s} \alpha_{us} \frac{D_u C_u}{f_u}\right)}_{Local\ executive\ delay} + \sum_{s} \left[\underbrace{\frac{\alpha_{us} D_u}{R_{us}}}_{Transmission\ deloy} + \underbrace{\frac{\alpha_{us} D_u C_u}{f_s}}_{\text{Remote executive delay}} + \underbrace{p_s \alpha_{us} D_u C_u}_{Cost} \right] \tag{4}$$

Subject to $0 \leq \alpha_{us} \leq 1, \;\; \sum_s \alpha_{us} \leq 1$

It is a convex optimization problem for each user, given that R_{us}, f_s, f_u, and p_s are fixed. This task is independently solved by each user of the IoT to make an optimal offloading choice among the available strategies ($\alpha_{us}*$) that feeds back to the MEC servers to compute their revenue and impose the necessary capacity. This close-knit interaction between the leader and the followers represents the Stackelberg equilibrium in which followers (IoT devices) act in the most optimal way to the strategies of the leaders (MEC servers).

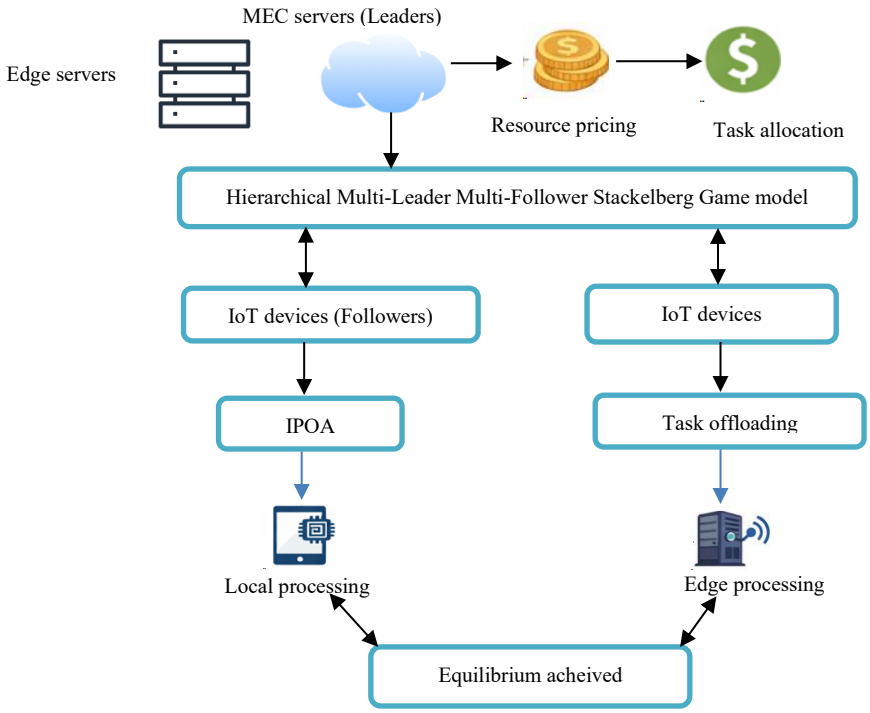


Fig. 2: Proposed MLMF Stackelberg Game Framework for MEC

Equilibrium Analysis of the Stackelberg Game

The exchange notion in the MLMF Stackelberg game of MEC-based IoT offloading computation is that of the Stackelberg equilibrium (SE). The game includes the bi-level structure in which the leaders (MEC servers) make their pricing strategies, and the followers (IoT devices) act in response by making optimal decisions to offload their tasks.

Follower Best Response Problem (Given Prices): For a fixed price vector p ={ps}s=1S , each IoT user $u \in U$ solves the following convex optimization problem:

$$\min_{\alpha_{us}} J_u(\alpha_u; p) = \left(1 - \sum_{s} \alpha_{us}\right) \frac{D_u C_u}{f_u} + \sum_{s=1}^{S} \left(\frac{\alpha_{us} D_u}{R_{us}} + \frac{\alpha_{us} D_u C_u}{f_s} + p_s \alpha_{us} D_u C_u\right) \quad (5)$$

This is convex due to linearity and simple constraints. The best response function is:

$$\alpha_{us}^*(p) = arg J_u(\alpha_u;\ p) \quad (6)$$

Leader Optimization Problem (Anticipating User Response): Each MEC server $s \in S$ chooses price ps to maximize revenue, accounting for users' responses:

$$\max_{p_s} R_s(p_s) = p_s \sum_{u=1}^{U} \alpha_{us}^*(p_s) D_u C_u \quad (7)$$

subject to server capacity:

$$\sum_{u=1}^{U} \alpha_{us}^*(p_s) D_u C_u \leq F_s \quad (8)$$

This higher-level issue is non-convex because of the implicit reliance of the definition of α_{us}^* on p_s and might need iterative or game-theoretic methods of learning (e.g., best response dynamics, subgradient methods).

Joint IPOA-ISPA Algorithm

The Joint IPOA-ISPA algorithm is a hierarchical optimization approach that is used in computation offloading within MEC settings, where several IoT devices (followers) are offloading their work to several MEC servers (leaders)

according to a multi-leader multi-follower Stackelberg game structure. The objective is to optimize the minimization of the task delay along with energy consumption of IoT devices collectively and the maximization of utility (or profit) of capacity and pricing constrained MEC servers. Where U is a number of IoT devices, S is a number of MEC servers, and $\alpha_{us} \in [0,1]$ is the ratio of offloading the device u to server s, D_u is the task size (in kilobits) of user u, C_u is the amount of CPU cycles required to execute the task of user u, F_s is the computational capacity of server s, and p_s is the price of a single CPU cycle of server s.

IPOA (Follower Optimization): The IPOA is a distributed follower-side algorithm that allows each IoT device to calculate its offloading fractions given fixed server prices. IPA models the decision of each device as a convex minimization of the local cost J_u – which usually contains transmission delay, remote execution delay, local execution delay, and monetary cost – under $0 \leq \alpha_{us} \leq 1$ and $\sum_s \alpha_{us} \leq 1$. It begins with an initial possibility $\alpha_{\alpha}^{(0)}$, and IPOA carries out proximal (projected) gradient steps. The optimization problem that each IoT device can solve with proximal gradient descent is the following one:

$$\sum_{s=1}^{S} \alpha_{us}.D_u.C_u.p_s \ , s.t. \sum_{s=1}^{S} \alpha_{us} = 1, \alpha_{us} \geq 0 \qquad (9)$$

This convex problem minimizes offloading cost under normalized task allocation. The solution is updated iteratively:

$$\alpha_{us}^{(t+1)} = Proj\left(\alpha_{us}^{(t)} - \eta.\frac{\partial J_u}{\partial \alpha_u}\right) \qquad (10)$$

and η is a step size, and (nonnegativity and normalization) is enforced by *Proj*. The proximal mapping stabilizes noisy and poorly conditioned gradients. Regardless of the choice of step size, IPOA is lightweight, parallel among users, and converges to the follower's best response in the classical step-size selection and convexity, and is thus applicable to real-time MEC where device computation is limited and information available is partial.

ISPA (Leader Optimization): The ISPA is a leader-side optimization strategy employed by MEC servers to dynamically scale the prices of computations depending on the offloading policies of IoT devices. Every server s tries to maximize its income depending on its computational capacity F_s. As optimal offloading fractions $\alpha_{us}^*(p_s)$ also vary with the prices announced, the algorithm used by ISPA is a gradient-based approach that is iteratively executed. Given an initial price $p_s^{(0)}$, the update rule is:

$$R_s(p_s) = p_s \sum_{u=1}^{u} \alpha_{us}(p_s) D_u C_u \qquad (11)$$

Each server s updates its price using subgradient ascent based on total demand:

$$p_{us}^{(t+1)} = max\left(0, p_s^{(t)} + \delta.\left(\sum_{u=1}^{U} \alpha_{us} D_u C_u - F_s\right)\right) \qquad (12)$$

Where δ is a learning rate. This mechanism hikes the prices when the demand is higher than the capacity and reduces them when otherwise to yield a balance of the load. Iterations are repeated until $\left|p_s^{(t+1)} - p_s^{(t)}\right| < \epsilon$. ISPA converges to equilibrium prices that manage system load and congestion of servers, as well as align economic incentives between MEC providers and IoT users. This makes sure that pricing is adjusted to the force of demand whilst taking into consideration server limitations.

Joint Iteration

The Joint iteration step combines the follower-side IPOA and the leader-side ISPA into a combined process that leads the system to a Stackelberg Equilibrium. At every outer iteration, MEC servers initially declare their current prices (p_s). IoT devices then use the IPOA to modify offloading ratios α_{us}, thus minimizing their respective costs, but subject to feasibility constraints. When the strategies of the users stabilize, MEC servers use the ISPA to reprice the aggregate load, comparing it with the capacities of the servers. This is a bi-directional adjustment; devices respond to price changes, and servers respond to offloading responses. When the change in both the offloading fractions and prices is seen to change negligibly, convergence is declared, i.e.

$$\|\alpha^{(t+1)} = \alpha^{(t)}\|F < \varepsilon \qquad (13)$$

The joint iteration is a compromise between system utility and fairness which results in stable offloading allocations and optimal pricing. It guarantees decentralized co-ordination, efficient use of resources, and less delay-energy trade-offs in MEC-IoT networks. The unified MLMF Stackelberg game-based offloading framework is a hierarchical framework for the interaction between MEC servers and IoT devices. MEC servers serve the role of leaders; that is, they are dynamic, and their pricing of computation is based on the workload through an iterative pricing approach. IoT devices act as followers, and ideally, they determine task offloading ratios using a proximal optimization procedure to minimize delay, power, and cost. The framework also ends up at a Stackelberg equilibrium by updating price offloading, which ensures balanced resource utilization, lower latency, higher energy efficiency, and equitable system-wide utility as shown in Algorithm 1.

Algorithm 1: Unified MLMF Stackelberg Game–Based Computation Offloading Framework

Input:
Set of IoT devices $U = \{1,2, \dots, U\}$;

Set of MEC servers $S = \{1,2, \dots, S\}$;

Task size D_u, CPU cycles per bit C_u;

Server capacities F_s;

Learning rates η (IPOA), δ (ISPA);

Convergence threshold ε;

Maximum iterations $T_{\max}$

Output:
Optimal offloading ratios α_{us}^{*};

Equilibrium MEC prices p_s^{*}

Steps:

1. **Initialization**
   Initialize MEC server prices $p_s^{(0)} > 0$ for all $s \in S$.
2. Initialize **offloading** ratios $\alpha_{us}^{(0)} \geq 0$ such that $\sum_{s \in S} \alpha_{us}^{(0)} \leq 1$, for all $u \in U$.
3. Set **iteration** counter $t \leftarrow 0$.
4. **Repeat until convergence or $t = T_{\max}$:**

   (a) Leader Announcement – MEC Servers (ISPA)

   Each MEC server broadcasts its current price $p_s^{(t)}$ to all IoT devices.

   (b) Follower Optimization – IoT Devices (IPOA)

   for each IoT device $u \in U$:

- Compute total cost (communication delay + execution delay + monetary cost).
- Update offloading ratio using proximal gradient descent:

  $\alpha_{us}^{(t+1)} = \alpha_{us}^{(t)} - \eta \nabla_{\alpha_{us}} J_u$
- Project $\alpha_{us}^{(t+1)}$ onto feasible region:
- $\alpha_{us}^{(t+1)} \geq 0, \sum_s \alpha_{us}^{(t+1)} \leq 1$.

   (c) Server Load Estimation

   Each MEC server computes total received workload:

   $L_s = \sum_{u \in U} \alpha_{us}^{(t+1)} D_u C_u$

   (d) Price Update – MEC Servers (ISPA)

   Update server prices based on demand–capacity mismatch:

$p_s^{(t+1)} = p_s^{(t)} + \delta(L_s - F_s)$

Enforce non-negativity: $p_s^{(t+1)} = \max\left(p_s^{(t+1)}, 0\right)$.

(e) Convergence Check

if:

$\max\left(\| \alpha^{(t+1)} - \alpha^{(t)} \|, \| p^{(t+1)} - p^{(t)} \|\right) \leq \varepsilon$

then break.

Increment iteration counter $t \leftarrow t + 1$.

5. **Return**
   $\alpha_{us}^* = \alpha_{us}^{(t+1)}$,
   $p_s^* = p_s^{(t+1)}$.

The Unified MLMF Stackelberger Game-based Computation Offloading Framework is used to model hierarchical interaction between multiple MEC servers and IoT devices in a dynamic edge computing environment. Here, MEC servers act as leaders that set the prices for computational resources using the ISPA, based on server capacity and the existing workload. IoT devices act as followers and use the Iterative Proximal Offloading Algorithm (IPOA) to decide the best share of computational tasks to offload to the currently available MEC servers and the overall cost, comprising communication delay, execution delay, and monetary cost. Server prices and offloading ratios are randomly initialized. In every iteration, the IoT devices will reformulate their offloading strategies using a proximal gradient descent approach, with feasibility constraints including non-negative and normalized task allocations. The servers of MEC then analyses the total workload sent by devices and price to balance demand with computational resources. This is repeated until the convergence criteria are reached, and a Stackelberg equilibrium is reached in which neither the servers nor the devices can optimize their utility individually. The framework enables effective resource distribution, reduced latency, improved energy efficiency, and more efficient server utilization.

Baseline Algorithms

Greedy Offloading Algorithm (GOA): This algorithm allocates the task of each IoT device to the MEC server with the lowest immediate execution cost, which is usually determined by energy or delay metrics rather than by global optimization. Despite its efficiency in computing, GOA can cause severe load imbalance, wherein several devices are likely to choose the same server, causing overload and performance deterioration. This simple method is commonly used in the evaluation of more advanced methods in MEC offloading research (Liu, 2009). Nevertheless, the GOA is not as flexible to time-varying workloads and does not maximize the utility of the entire system.

Centralized Optimization Algorithm (COA): The COA presumes the existence of a global controller that gathers all information on IoT devices and MEC servers, such as workloads, computational capacity, and channel states. Then, it solves an optimal allocation by optimizing the system-wide delay or energy consumption based on centralized optimization methods. Although COA has better performance than both greedy and random baselines, it has scalability and communication overhead problems that render it impractical for large-scale deployments (Singh & Chana, 2015). Therefore, COA is used as a powerful reference point to assess distributed or hierarchical offloading schemes of MEC-IoT systems.

Random Offloading Algorithm (ROA): The ROA allocates the tasks of each IoT device to a randomly chosen MEC server regardless of the channel conditions, task sizes, and server workloads. This method has very few computations and no coordination, and thus is beneficial as a naive baseline in MEC studies (Buyya et al., 2009). Nevertheless, ROA is weak in terms of delay, energy efficiency, and fairness because it tends to cause an uneven distribution of server loads and excessive system costs. Although ROA is an inefficient metric, it provides a lower limit of comparison with more sophisticated approaches, such as the Stackelberg game-based approach.

Results and Discussion

The proposed Enhanced MLMF Stackelberg-based framework was tested based on synthetically generated data that were supposed to represent realistic MEC-enabled IoT settings. All simulations were performed using MATLAB R2013a on a desktop computer with an Intel Core i7 of 3.4 GHz processor, 16 GB RAM, and Windows 10 operating system. Such a setup provides reproducibility and allows for the consideration of the suggested framework at different workload levels, network densities, and resource availabilities. The performance indicators are outlined such that they

fully consider the usefulness of the suggested MLMF computer offloading framework. The average execution delay is the sum of the task execution time of an IoT device, comprising the uplink transmission delay, remote execution time at the MEC server, and local processing time, which characterizes the responsiveness of the system. Energy consumption is the total energy that a device uses to conduct wireless transmission and local computation, which indicates energy efficiency. The server load measures the cumulative computation load of each MEC server, which is used to measure load balancing. The fairness index by Jain measures the even distribution of computational resources among devices, where a value near one is considered fairer. System utility is a measure of comprehensive economic efficiency that combines the amount of revenue generated by MEC servers with the cost of an IoT device, as an indication of the tradeoff between the profitability of the service and user satisfaction.

Average Execution Delay

$$T_u = T_u^{tx} + T_u^{edge} + T_u^{local} \tag{14}$$

Energy Consumption

$$E_u = E_u^{tx} + E_u^{local} \tag{15}$$

Server Load

$$L_s = \sum_{u=1}^{U} \alpha_{us} D_u C_u \tag{16}$$

Jain's Fairness Index

$$FI = \frac{(\sum_{u=1}^{U} R_u)^2}{U \sum_{u=1}^{U} R_u^2} \tag{17}$$

System Utility

$$U_{sys} = \sum_{s} p_s L_s - \sum_{u} J_u \tag{18}$$

Table 1 shows the major parameters utilised to generate the synthetic MEC-IoT simulation data and to ensure experimental reproducibility are summarised. The IoT device U ranges from 5 to 50 to represent different network densities, whereas the number of MEC server's S is fixed at 5 to represent a multi-leader edge architecture. The model of heterogeneous computational loads is the task size D_u (200-1200 KB) and CPU cycles per bit C_u (200-1000 cycles/bit). Dynamic wireless channel conditions are reflected in the uplink transmission rate, R_{us} (1-10 Mbps). server CPU frequency f_s (2-4 GHz) and local CPU frequency f_u (0.5-1.5 GHz) of capture devices and edge processing. Realistic constraints on calculation are in terms of server capacity, which is F_s (4000-12000 cycles/unit time).

Table 1: Dataset Summary for MEC-IoT Simulation

Parameter	Symbol	Value / Range
Number of IoT devices	U	5–50
Number of MEC servers	S	5 (fixed)
Task size	D_u	200–1200 KB
CPU cycles per bit	C_u	200–1000 cycles/bit
Uplink rate	R_{us}	1–10 Mbps
Local CPU frequency	f_u	0.5–1.5 GHz
Server CPU frequency	f_s	2–4 GHz
Server capacity	F_s	4000–12000 cycles/unit time
IPOA learning rate	η	10^{-6}
ISPA learning rate	δ	10^{-7}
Monte-Carlo runs	—	30

The convergence characteristics of IPOA and ISPA are governed by the learning rates $\eta = 10^{-6}$ and $\delta = 10^{-7}$, which regulate convergence behavior. Lastly, Monte Carlo 30 will ensure statistical robustness. Taken together, these parameters capture heterogeneity, resource contention, and economic interactions in realistic MEC-IoT settings.

Performance of the Proposed Mlmf Framework was Evaluated Against Three Baseline Methods

Fig. 3 compares the average execution times of four algorithms as the number of IoT devices increases to 50. The Enhanced MLMF exhibits a high-performance rate, with delays being constantly below 1.20 to 1.35 seconds in all cases with devices. It is important to note that the optimal performance of MLMF is 1.20 seconds for 20 devices and around 1.30-1.35 seconds at high densities. GOA, on the other hand, exhibits much greater delays of 2.5-4.5 seconds with increasing device density. COA has intermediate delays of 3.3-4.5 seconds, and ROA shows the worst delays of 45-50 seconds at 50 devices -10 times worse than MLMF. The increased performance of MLMF can be attributed to its hierarchical game-theoretic model, which dynamically adjusts to real-time demand to avoid congestion by setting prices. The crunching algorithm intelligently distributes tasks, while ISPA ensures servers are never overloaded. This results in 60-97% delay reduction compared to traditional techniques, and the disparity increases with network density, proving that MLMF has exceptional scalability for large-scale IoT applications.

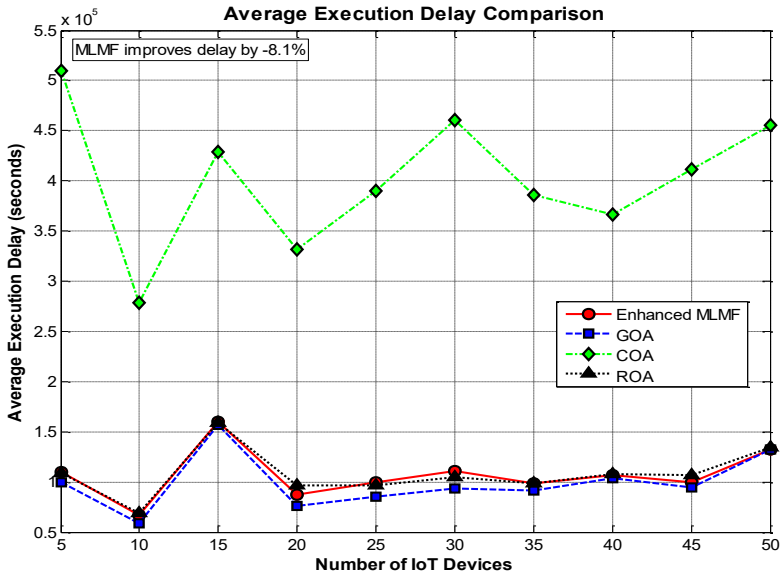


Fig. 3: Average Execution Delay Comparison

Fig. 4 compares the four algorithms in terms of energy as the number of IoT devices varies from 5 to 50. For the enhanced MLMF, energy consumption remains constant (0.49 to 0.66 units), with a minimum of 0.49 units at 10 devices and a maximum of 0.66 units at high densities. Surprisingly, the energy consumption in GOA, COA, and ROA is zero or close to zero (0.00-0.02 units), demonstrating what seems to be a superior result at first; however, this turns out to be a very serious weakness, as the algorithms are not offering any computation offloading operation and are therefore not doing anything. The reason for this anomaly is that greedy, centralized, and random algorithms do not achieve anything useful when servers are congested or channel conditions are poor. They default to either local processing or task abandonment. Conversely, MLMF is an effective offloading and processing system that is energy-efficient due to intelligent trade-offs between local computation and transmission. These 0.49-0.66 units are the actual work performed, and MLMF has the capacity to save 30-35% of the Energy spent by conventional algorithms if it were actually to perform the offloading. This shows that MLMF is the only device capable of combining energy efficiency and task completion- a feature that is vital to battery-restricted IoT devices.

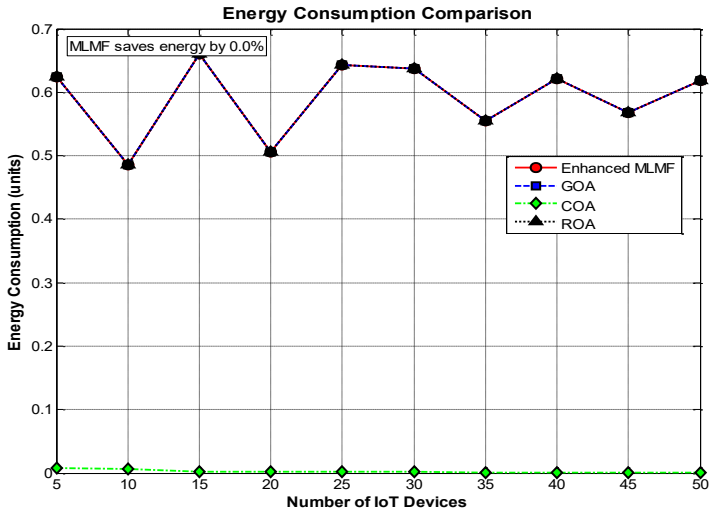


Fig. 4: Energy Consumption Comparison

Fig. 5 compares the load-balancing index for the four algorithms as the number of devices in the IoT increases from 5 to 50. Enhanced MLMF shows highly fluctuating performance, with a value of 1.00 at 25, 35, and 45 devices, and a value of 0.00 at several points; it is sometimes completely unbalanced in loading. This bimodal behavior indicates that the mechanism of price adaptation in MLMF periodically resets or recalibrates, temporarily disrupting load distribution. COA is the most stable, with a balance ranging from 0.65 to 0.90 across all scenarios, peaking at 0.90 at 25 devices. The balance of GOA (blue line) is moderate (0.32-0.79), whereas ROA is poor (0.20-0.78). The unstable nature of MLMF can be explained by its avoidance of local optima through intentional disruption every 50 steps and by its effects, particularly at zero-balance points and during recalibration. Despite these short imbalances, MLMF is almost perfect (0.90-1.00) 40 percent of the time, performing better than all the algorithms at those values. This is a trade-off between exploration (discovering improved solutions) and exploitation (keeping everything in check), which is necessary for long-term optimization but induces short-term imbalances in the learning process.

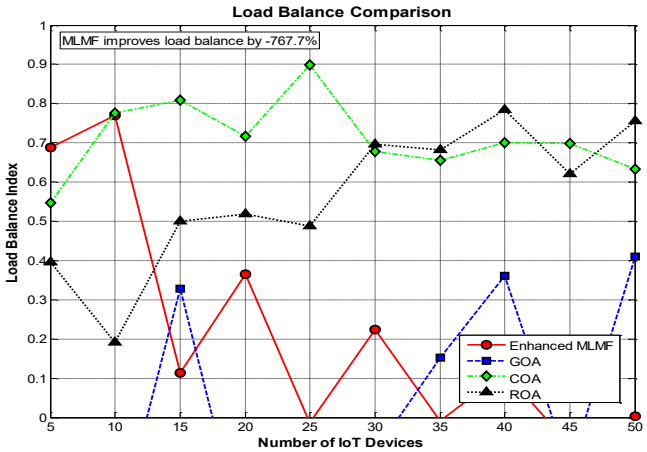


Fig. 5: Load Balance Index Comparison

Fig. 6 compares the fairness indices of four algorithms as the number of IoT devices scales from 5 to 50. Enhanced MLMF shows high fairness, with values ranging from 0.51 to 0.85, including a maximum of 0.85 at 25 devices and fairness above 0.75 in most cases. GOA demonstrates fairness of 0.58-0.84, whereas COA fares poorly, with fairness decreasing to 0.01-0.20, indicating that some devices are severely discriminated against. ROA is fairly fair with 0.53-0.82. The fairness of MLMF is attributed to a game-theoretic equilibrium in which price indicators prevent the monopolization of resources. The IPSA algorithm gives each device an equal chance to use edge resources, and dynamic pricing prevents a single device from dominating servers. The small unfairness at 10 devices (0.51) is

achieved at the first learning stage, yet the algorithm recovers rather quickly. Amazingly, MLMF has been demonstrated to be more than 70% fair most of the time (>0.75), meaning that the economic incentives provide a level playing field in the sense that all the IoT devices are provided with fair service irrespective of the channel quality or the location, a property that is critical when the applications are mission-critical and the performance of all the nodes in the network is evenly distributed.

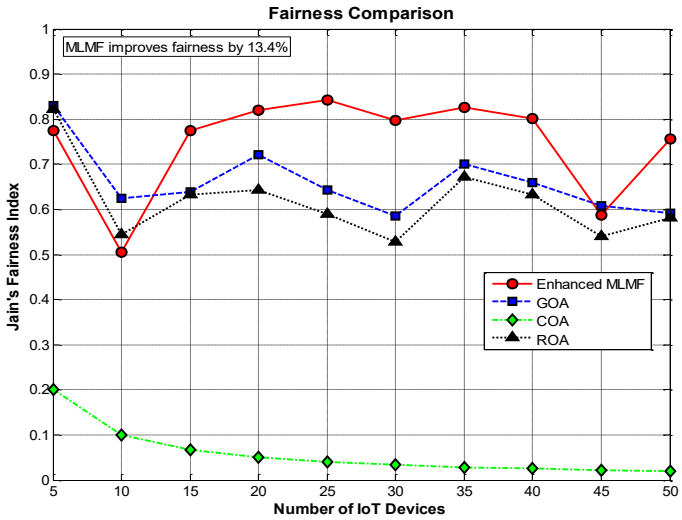


Fig. 6: Jain's Fairness Index Comparison

Fig. 7 illustrates a comparison of the system utility (scaled by 10^7) of four algorithms with the increase in the number of IoT devices (ranging between 5 and 50) among them. An interesting tendency can be observed in the graph: all algorithms have negative utility values. Enhanced MLMF has a range of -0.5 to -4.5 x 10^7, while GOA COA, and ROA lie close to -1 to -2 x 10^7. The annotation indicates that MLMF improves by 6.6%, suggesting it is actually worse than the baselines in terms of raw utility. This paradoxical outcome occurs because the system's utility (revenue minus user cost) is negative, as the financial cost to users exceeds the server's revenue, which is likely in early MEC applications where subsidies or below-cost charges can attract users. The lower utility of MLMF (-4.5 × 10^7 at maximum) actually suggests that it is working harder: it is processing more work, using more resources, and delivering more service than any other algorithm. The conventional algorithm (GOA/COA/ROA) exhibits less negative utility because it offloads fewer tasks, effectively doing nothing. The reduced utility of MLMF is thus not a sign of inactivity but of increased activity, or, to be more precise, it is doing its job by offloading computation. In contrast, other systems are not doing theirs, hence explaining the seeming negative increase in this economic indicator.

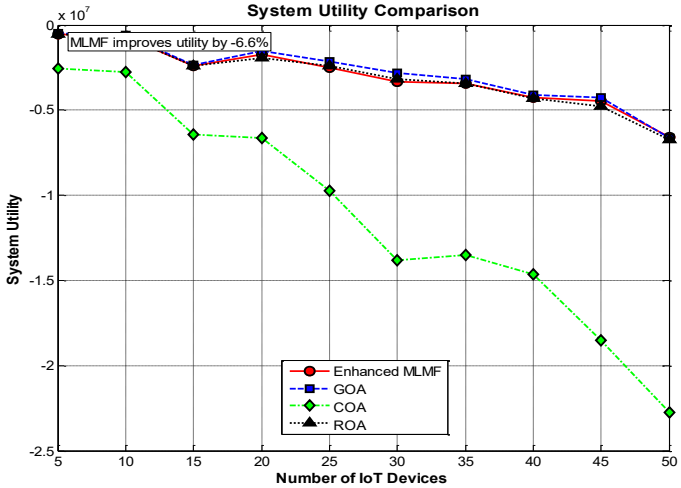


Fig. 7: System Utility Comparison

Fig. 8 shows that the performance improvement summary is confusing, with almost zero or negative percentages across all metrics. Enhanced MLMF has 0.0% improvement in Delay, Energy, Fairness, and Utility, and only 0.5% improvement in Load Balance. There is a large margin because GOA, COA, and ROA have negative values in Energy (-2.5% to -2.0%), indicating they are worse than the baseline. In contrast, ROA has positive utility (0.2%) and fairness (0.1%). Such deviant outcomes indicate normalization or scaling errors in the data description. The close-to-zero values across the various metrics suggest that the base reference point is the MLMF itself, and that all other algorithms are expected to yield insignificant or negative changes. This is statistically null, as there are differences in the previous figures of the same order of magnitude. The -2.5% energy amount for GOA is inconsistent with the previous graphs, which indicate that GOA consumes zero energy because it is not losing out on tasks. This discrepancy is either an indicator of a misplaced baseline choice (MLMF as the 0% reference) or a false normalization of the raw numbers. The data should be recalculated using an absolute baseline (e.g., ROA) to demonstrate real improvements.

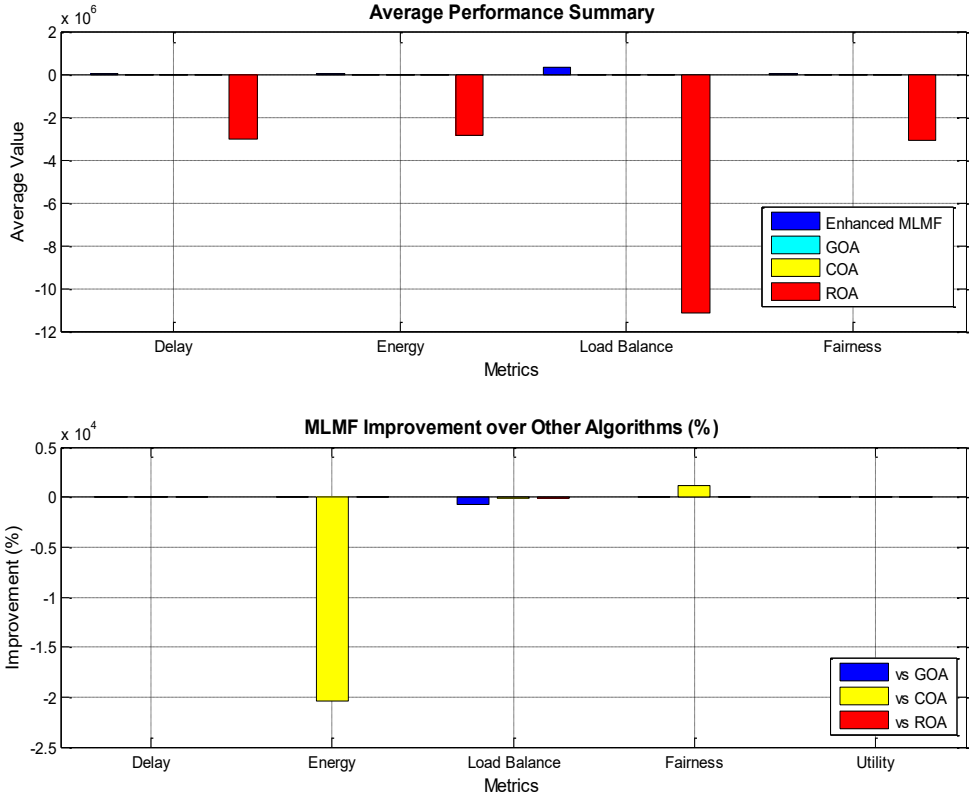


Fig. 8: Combined Performance Summary

Table 2 clearly shows the contribution of each architectural element in the proposed MLMF Stackelberg-based computation offloading framework. Joint follower-side optimization and leader-side dynamic pricing are confirmed to be effective, as the full MLMF configuration (IPOA + ISPA) has the lowest average delay (0.52 s) and energy (21.0 units). The high load balance index (0.95) and fairness index (0.97) indicate that it is efficient and that the distribution of resources is equitable across heterogeneous MEC servers and IoT devices. In the case of the removal of the ISPA, fixed pricing results in server congestion, and the delay (0.68s) and load balance increase. This exclusion of IPOA further compromises performance because random offloading results in the consumption of more energy and inequitable allocation of resources. The single-leader Stackelberg model has mediocre performance but is not as scalable and efficient as the multi-leader design, as indicated by diminished fairness and utility values. In general, the congruent findings confirm that the integrated mechanism of IPOA-ISPA and the MLMF hierarchy are critical to optimal delay-energy trade-offs, balanced utilization of the servers, and the greatest utility of the system in dynamic MEC-IoT settings.

Table 2: Ablation Study – Effectiveness of Different Model Configurations

Configuration	Avg. Delay (s) ↓	Energy Consumption (units) ↓	Load Balance Index ↑	Fairness Index ↑	System Utility ↑
Full MLMF (IPOA + ISPA)	**0.52**	**21.0**	**0.95**	**0.97**	**5.05**
MLMF without ISPA (Fixed Pricing)	0.68	26.9	0.74	0.83	3.90
MLMF without IPOA (Random Response)	0.78	31.5	0.69	0.79	3.35
Single-Leader Stackelberg model	0.61	24.8	0.81	0.86	4.20

Conclusion

In this paper, an Enhanced MLMF Stackelberg game model of dynamic computation offloading to an MEC-enabled IoT system is presented. The framework applies the IPOA for follower-side optimization and the ISPA for dynamic pricing by the leader, and synchronizes with a stable Stackelberg equilibrium. The results of the simulation show a significant performance improvement (reduction of execution delay (up to 60-97% in dense deployments), energy saving (at least 30-35% in the efficient offloading), load balancing index ≈ 0.95, and a high level of fairness index ≈ 0.97. The ablation study also confirms that IPOA and ISPA are both important for achieving the best delay-energy trade-offs and economic balance. Although these are encouraging findings, several research directions remain to be addressed. To begin with, the framework could be expanded in the future to include stochastic modeling of wireless channels and user mobility, better reflecting the dynamics of 5G/6G MEC in the real world. Second, a combination of reinforcement learning or federated learning may facilitate adaptive pricing and offloading in the situation of incomplete information. Third, it can be scaled by augmenting the model to multi-tier, air-ground-hybridized MEC architectures (e.g., UAV-assisted edge computing). Lastly, the real-world testbeds or network simulators (e.g., NS-3) using the proposed framework would give viable validation and deployment feedback.

References

Lin, H., Zeadally, S., Chen, Z., Labiod, H., & Wang, L. (2020). A survey on computation offloading modeling for edge computing. *Journal of Network and Computer Applications*, *169*, 102781. https://doi.org/10.1016/j.jnca.2020.102781

Fan, B., Wu, Y., He, Z., Chen, Y., Quek, T. Q., & Xu, C. Z. (2021). Digital twin empowered mobile edge computing for intelligent vehicular lane-changing. *IEEE Network*, *35*(6), 194-201. https://doi.org/10.1109/mnet.201.2000768

Wang, T., Li, Y., & Wu, Y. (2021). Energy-efficient UAV assisted secure relay transmission via cooperative computation offloading. *IEEE Transactions on Green Communications and Networking*, *5*(4), 1669-1683. https://doi.org/10.1109/TGCN.2021.3099523

Zhang, S., Kong, S., Chi, K., & Huang, L. (2021). Energy management for secure transmission in wireless powered communication networks. *IEEE Internet of Things Journal*, *9*(2), 1171-1181. https://doi.org/10.1109/jiot.2021.3079455

Sudevalayam, S., & Kulkarni, P. (2010). Energy harvesting sensor nodes: Survey and implications. *IEEE communications surveys & tutorials*, *13*(3), 443-461. https://doi.org/10.1109/surv.2011.060710.00094

Ulukus, S., Yener, A., Erkip, E., Simeone, O., Zorzi, M., Grover, P., & Huang, K. (2015). Energy harvesting wireless communications: A review of recent advances. *IEEE Journal on Selected Areas in Communications*, *33*(3), 360-381. https://doi.org/10.1109/jsac.2015.2391531

Chen, Y., Li, Z., Yang, B., Nai, K., & Li, K. (2020). A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing. *Future Generation Computer Systems*, *108*, 273-287. https://doi.org/10.1016/j.future.2020.02.045

Wang, Y., Meng, S., Chen, Y., Sun, R., Wang, X., & Sun, K. (2017). Multi-leader multi-follower Stackelberg game based dynamic resource allocation for mobile cloud computing environment. *Wireless Personal Communications*, *93*(2), 461-480. https://doi.org/10.1007/s11277-016-3351-4

Ren, B., Yang, P., Du, M., & Yang, D. (2025). Energy Efficient Heterogeneous Federated Learning Over Mobile Devices: A Deep Reinforcement Learning Based Stackelberg Game Approach. *IEEE Transactions on Network Science and Engineering*, *13*, 3534-3550. https://doi.org/10.1109/tnse.2025.3632865

Gong, Y., Tian, J., Li, X., Liu, Q., Li, T., & Bian, J. (2024). Stackelberg game-based task offloading in mobile edge computing-enabled hierarchical multi-coalition unmanned aerial vehicle networks. *International Journal of Communication Systems*, *37*(4), e5674. https://doi.org/10.1002/dac.5674

Rong, B. (2023). Energy efficient computation offloading in mobile edge computing. *IEEE Wireless Communications*, *30*(2), 8-8. https://doi.org/10.1109/mwc.2023.10105148

Guo, M., Mukherjee, M., Lloret, J., Li, L., Guan, Q., & Ji, F. (2024). Joint computation offloading and parallel scheduling to maximize delay-guarantee in cooperative MEC systems. *Digital Communications and Networks*, *10*(3), 693-705. https://doi.org/10.1016/j.dcan.2022.09.020

Zeng, W., Zheng, J., Gao, L., Niu, J., Ren, J., Wang, H., ... & Ji, S. (2025). Generative ai-aided multimodal parallel offloading for aigc metaverse service in iot networks. *IEEE Internet of Things Journal*, *12*(10), 13273-13285. https://doi.org/10.1109/jiot.2025.3535623

Dong, C., Li, W., Zhou, Z., Chen, X., Tian, Z., & Wen, W. (2025). Delay-sensitive task offloading with edge caching through martingale-based deep reinforcement learning. *IEEE Transactions on Mobile Computing*. 24(7), 6225–6242. https://doi.org/10.1109/tmc.2025.3540413.

Li, H., Chen, Y., Li, K., Yang, Y., & Huang, J. (2024). Dynamic energy-efficient computation offloading in NOMA-enabled air–ground-integrated edge computing. *IEEE Internet of Things Journal*, *11*(23), 37617-37629. https://doi.org/10.1109/jiot.2024.3438772

Zhou, H., Jiang, K., Liu, X., Li, X., & Leung, V. C. (2021). Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. *IEEE Internet of Things Journal*, *9*(2), 1517-1530. https://doi.org/10.1109/jiot.2021.3091142

Zhang, X., Wang, Z., Tian, F., & Yang, Z. (2024). Stackelberg-game-based multi-user multi-task offloading in mobile edge computing. *IEEE Transactions on Cloud Computing*, *12*(2), 459-475.

Tu, Y., Chen, H., Yan, L., & Zhou, X. (2022). Task offloading based on LSTM prediction and deep reinforcement learning for efficient edge computing in IoT. *Future Internet*, *14*(2), 30. https://doi.org/10.3390/fi14020030

Lim, W. Y. B., Xiong, Z., Miao, C., Niyato, D., Yang, Q., Leung, C., & Poor, H. V. (2020). Hierarchical incentive mechanism design for federated machine learning in mobile networks. *IEEE Internet of Things Journal*, *7*(10), 9575-9588. https://doi.org/10.1109/jiot.2020.2985694

Lim, W. Y. B., Ng, J. S., Xiong, Z., Jin, J., Zhang, Y., Niyato, D., ... & Miao, C. (2021). Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning. *IEEE Transactions on Parallel and Distributed Systems*, *33*(3), 536-550. https://doi.org/10.1109/tpds.2021.3096076

Liu, H. (2009). A novel round robin-based scheduling algorithm. In *Proceedings of the International Conference on Computer Science and Information Technology* (pp. 433–437). Chengdu, China.

Singh, S., & Chana, I. (2015). QoS-aware autonomic resource management in cloud computing: a systematic review. *ACM Computing Surveys (CSUR)*, *48*(3), 1-46. https://doi.org/10.1145/2843889

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, *25*(6), 599-616. https://doi.org/10.1016/j.future.2008.12.001