LEAOT: Load-, Deadline-, Latency-, and Energy-Aware Task Offloading for Scalable IoT Edge–Cloud Systems

Ayman Noor
Department of Computer Science-College of Computer Science and Engineering, Taibah University, Madinah 42353, Saudi Arabia

***Abstract*—Internet of Things (IoT) applications increasingly offload sensing and analytic tasks to edge and cloud resources. Cloud processing offers high computational capacity but can increase round-trip delay and wireless energy consumption, while edge processing reduces access delay but can suffer from queue buildup when many devices select the same nearby node. This study proposes *LEAOT*: Load-, Deadline-, Latency-, and Energy-Aware Task Offloading for Scalable IoT Edge–Cloud Systems, a lightweight load- and deadline-aware task offloading method for IoT edge–cloud systems. Unlike black-box learning methods, *LEAOT* uses an explainable online score that combines the estimated communication delay, the First-Come First-Served (FCFS) aggregate queue delay, the execution delay, the device-side energy, the soft deadline pressure, and a dimensionless infrastructure-pressure term. The method also uses an exponentially weighted link estimate and a virtual workload correction step so that stale bandwidth and queue estimates are not treated as fixed constants. A controlled discrete-event evaluation is conducted across 10 independent trials, heterogeneous task sizes, and scalable edge topologies ranging from 2 to 16 edge nodes. Results show that *LEAOT* reduces deadline violation to 0.69% and maintains low energy of 0.117 J/task under the reference setting, while remaining competitive with delay-resource and drift-plus-penalty baselines. The results also quantify the effect of node scalability and the resource-pressure weight.**

***Keywords*—Internet of Things; edge computing; cloud computing; task offloading; latency; energy efficiency; deadline-aware scheduling**

I. Introduction

Internet of Things (IoT) systems collect data from cameras, wearable sensors, meters, industrial controllers, vehicles, and environmental monitoring devices [1], [2]. Many of these devices cannot process all generated tasks locally due to limited battery capacity and Central Processing Unit (CPU) resources [3], [4]. Cloud computing can handle heavy workloads, but remote cloud data centers are often accessed via longer network paths. This may increase round-trip latency, backhaul use, and transmit energy. Edge computing moves computation closer to the data source, thereby reducing access delay, but the edge layer has limited processing capacity and may become congested when many tasks are routed to the same node [5], [6], [7].

The offloading decision is therefore not a simple edge vs. cloud choice. A gateway must consider communication time, current node workload, task size, required CPU cycles, wireless energy, and deadline urgency [8], [9]. Recent studies have used optimization, pricing, matching, reinforcement learning, and deep learning to address such problems [10], [11], [12], [13]. These approaches are valuable, but some require training, global state, iterative solvers, or careful parameter tuning. In small and medium IoT deployments, a transparent online method remains useful because a gateway may need to make a quick decision based solely on current measurements.

Consider a smart city scenario in which hundreds of surveillance cameras, wearable health devices, industrial sensors, and autonomous robots perform computation-intensive tasks with varying latency and deadline constraints at a high rate. Under normal circumstances, edge servers with low communication latency serve as the preferred execution destinations [14], [15]. If multiple neighboring devices rush to offload tasks to the same edge server, though, waiting times will increase dramatically, leading to deadline misses irrespective of the short communication distance [16], [17]. Offloading to the cloud, while relieving pressure on local compute nodes, comes with significantly greater processing capacity at the expense of higher communication latency and increased wireless energy drain on battery-operated devices. As a result, edge-centric or cloud-centric heuristics often yield suboptimal results under fluctuating workloads.

The challenge thus arises of deciding, upon each task's arrival, where to execute it (i.e., locally at an edge node or in the cloud) to optimize multiple criteria. These can include, but are not limited to, expected execution latency and wireless energy usage, node workload, expected deadline miss rate, and resource utilization. Given the ever-changing system state (e.g., available bandwidth and queue sizes), threshold-based schemes are hard to calibrate, and heavyweight optimization frameworks are ill-suited for massive IoT networks. Instead, an online task offloading policy capable of consistently adapting to the current system state is required.

To address this challenge, this study proposes the *Load-, Deadline-, Latency-, and Energy-Aware Task Offloading Technique* (*LEAOT*). The method is intentionally learning-free, but it is not presented as a basic weighted sum. Its contribution is an explainable online decision kernel that explicitly combines queue correction, link-state smoothing, soft-deadline pressure, and device-energy estimation into a single gateway-level algorithm. The design is useful when the gateway has periodic estimates of edge load and bandwidth but lacks sufficient historical data to train a robust learning model.

The contributions are summarized as follows:

- A load-aware IoT edge–cloud system model is developed with explicit First-Come First-Served (FCFS) aggregate queue assumptions, Exponentially Weighted Moving Average (EWMA) bandwidth estimation, and device-side energy accounting.
- *LEAOT* is proposed as a lightweight online task offloading model that jointly considers virtual queue-workload correction, EWMA-based channel bandwidth prediction, edge-cloud queue-aware task latency modeling, on-device energy consumption modeling, soft deadline-based task urgency evaluation, and edge-cloud infrastructure congestion awareness as part of a holistic normalized decision scoring system.
- The evaluation uses ten independent trials, 95% confidence intervals, and scalable node counts rather than a fixed two- or three-node topology.
- *LEAOT* is compared with practical and literature-inspired baselines, including queue-aware edge–cloud selection, threshold selection, delay-resource-cost scoring, and drift-plus-penalty scoring.

The rest of this study is structured as follows. Section II provides an overview of recent efforts towards IoT task offloading across the edge–cloud computing continuum and highlights the research gap this study attempts to fill. Section III introduces the *LEAOT* model. First, the system architecture is described. Then the mathematical formulation is presented, followed by the models used to compute latency and energy consumption, and the formulation for workload estimation. Then, the online task offloading algorithm is provided. Section IV details the simulation setup, including workload generation, baseline methods, evaluation metrics, and experimental configurations. Section V presents and discusses the experimental results, followed by comparative analysis, sensitivity analysis, and scalability tests under different edge cluster sizes, and discusses the proposed model limitations. Section VI concludes this study with a discussion of the findings, limitations, and future work.

II. Related Work

Task offloading is an essential research topic in Mobile Edge Computing (MEC), fog computing, and cloud-edge collaborative systems. Recent survey papers have categorized offloading strategies based on latency, energy efficiency, Quality of Service (QoS), authentication, resource management, and application-specific perspectives [10], [18]. They both concluded that, in practice, offloading decisions should consider multiple objectives.

Recently, several heuristic and optimization-based task-offloading solutions have been proposed for edge-cloud systems. An energy-aware secure task offloading framework for multi-tier edge–cloud systems that jointly considers security and energy consumption was proposed in [19]. Du *et al.* proposed a novel Q-learning-based load-balancing scheme that offloads computation workloads between edge/cloud servers adaptively for lower response time [20]. Liu *et al.* presented fast collaborative edge–cloud task offloading with minimized delay and resource costs subject to reliability [11]. The above works have shown that delay, workload balancing, and resource utilization should be jointly considered in online scheduling.

Some recent works have also explored the economic/service-based incentives for offloading. Li *et al.* [12] presented a tiered-pricing mechanism for collaborative edge–cloud computing, which models service pricing as part of a matching-game formulation when making task allocations. Li *et al.* [21] leverage ideas from Service Level Agreements (SLAs) and imposed SLA constraints in fog computing workflows while minimizing energy consumption under a deadline-aware task execution requirement. However, these methods rely on provider-specific pricing knowledge or workflow assumptions that do not generalize well across most settings. For this reason, the proposed *LEAOT* approach features a dimensionless infrastructure-pressure heuristic that penalizes offloading computation to the cloud while avoiding cloud-provider-specific billing plans.

Methods that leverage learning-based heuristics have gained significant interest in recent years due to their adaptive nature to shifting workloads. Wang *et al.* [22] designed a deep reinforcement learning scheduler to jointly optimize system load and task response time in edge/fog computing systems. Benaboura *et al.* [13] utilized a Deep Q-Network (DQN) for decision-making to minimize overall latency and energy spent on devices deployed in IoT–cloud systems. Bui and Yoo [23] leveraged interruption-aware offloading for Industrial IoT applications, accounting for varying computation resource availability that can cause abrupt service interruptions. However, learning-based approaches entail the cost of acquiring training datasets, hyperparameter tuning, model maintenance, and retraining models as the workload changes.

Dai *et al.* introduced a federated deep reinforcement learning framework that enables distributed agents to collaboratively improve task offloading decisions while maintaining a decentralized learning manner [24]. While the proposed framework enhances scalability and flexibility, it still inherits the challenges of distributed model training and synchronization among collaborative edge nodes. To address these issues, Chen *et al.* proposed a digital twin-assisted multi-agent reinforcement learning framework for decentralized collaborative edge computing that enables distributed task offloading in an agent-driven paradigm without a central coordinator node [25]. Despite making progress towards decision-making robustness in distributed environments, continual policy learning and coordination are required among intelligent agents.

In contrast to prior learning-based approaches, the proposed *LEAOT* algorithm performs lightweight online task scheduling without requiring iterative optimization or model training. By aggregating queue-corrected latency estimates, device-side energy consumption, deadline pressure, and infrastructure pressure into a normalized multi-objective score, *LEAOT* offers an explainable and computationally efficient decision-making mechanism suitable for real-time edge gateways.

Table I summarizes the representative task offloading schemes reported in the recent literature. Briefly, existing works fall into three categories: survey papers, optimization-based works, and learning-based works. Survey works typically give detailed taxonomies and list open problems but do not provide concrete methodologies to address the problems;

TABLE I. COMPARISON OF RECENT TASK OFFLOADING APPROACHES AND THE PROPOSED *LEAOT* MODEL

Ref.	Main Objective	Decision Strategy	Learning-Based	Key Difference from LEAOT
[11]	Delay and resource-cost optimization	Collaborative optimization	No	Uses optimization-based resource allocation, whereas LEAOT employs a computationally efficient normalized scoring mechanism.
[12]	Pricing-aware edge–cloud offloading	Matching-game optimization	No	Depends on provider-specific pricing models, while LEAOT introduces a provider-independent infrastructure-pressure metric.
[13]	Latency- and energy-aware task offloading	Deep Q-Network (DQN)	Yes	Relies on neural-network inference, while LEAOT performs transparent score-based scheduling with lower computational overhead.
[19]	Secure and energy-aware task offloading	Multi-tier optimization	No	Emphasizes security and energy efficiency, whereas LEAOT focuses on lightweight real-time scheduling with multiple performance objectives.
[20]	Real-time workload balancing	Q-learning	Yes	Requires policy learning and convergence before deployment, while LEAOT performs immediate online decisions without training.
[21]	Energy-efficient SLA-aware workflow scheduling	Workflow optimization	No	Designed for workflow scheduling under SLA constraints rather than task-level online IoT offloading.
[22]	Load balancing and response-time optimization	Deep Reinforcement Learning	Yes	Requires model training, hyperparameter tuning, and continuous updates, whereas LEAOT is fully training-free and interpretable.
[23]	Interruption-aware Industrial IoT offloading	Resource-aware scheduling	No	Targets Industrial IoT interruption scenarios, whereas LEAOT is designed as a general-purpose edge–cloud offloading framework.
[24]	Digital twin edge task offloading	Federated Deep Reinforcement Learning	Yes	Requires distributed model training and synchronization, whereas LEAOT performs lightweight online scheduling without learning.
[25]	Decentralized collaborative edge offloading	Multi-agent Reinforcement Learning	Yes	Targets decentralized coordination among intelligent agents, while LEAOT employs a centralized gateway-level heuristic with low computational overhead.
Proposed LEAOT	Latency-energy-aware online task offloading	Normalized multi-objective heuristic	No	Integrates queue-corrected latency, device energy, deadline pressure, and infrastructure pressure into a lightweight, scalable, interpretable, and training-free online decision framework.

optimization-based works seek to optimize latency, energy, resource utilization, price or service-level goals through various optimization formulations; learning-based works adapt well to dynamically changing environments but require training procedures and hyperparameter tuning/jitter as well as model updating procedures. Compared with these works, the proposed *LEAOT* is a lightweight, interpretable, training-free online decision rule that combines queue-corrected latency prediction with device-side energy consumption costs, deadline pressure, and infrastructure pressure into a single normalized score function, facilitating computationally efficient online task scheduling that scales across heterogeneous edge–cloud systems.

III. SYSTEM MODEL AND PROPOSED TECHNIQUE

A. Edge–Cloud Architecture

Fig. 1 depicts an overview of the proposed three-layer task offloading framework consisting of IoT Layer, Edge Layer, and Cloud Layer seamlessly connected through the centralized Gateway Broker. The Gateway Broker centrally runs the proposed *LEAOT* scoring engine. Each layer is described below.

The IoT Layer consists of various types of sensors and resource-constrained IoT devices that initiate tasks with unique features, such as input data size, required CPU cycles, and time deadlines. When a task request arrives at the Gateway Broker, all associated metrics from the execution environment are collected periodically throughout execution. These runtime features can include each target's available link bandwidth (i.e., estimated using Weighted Moving Average (EWMA)), node

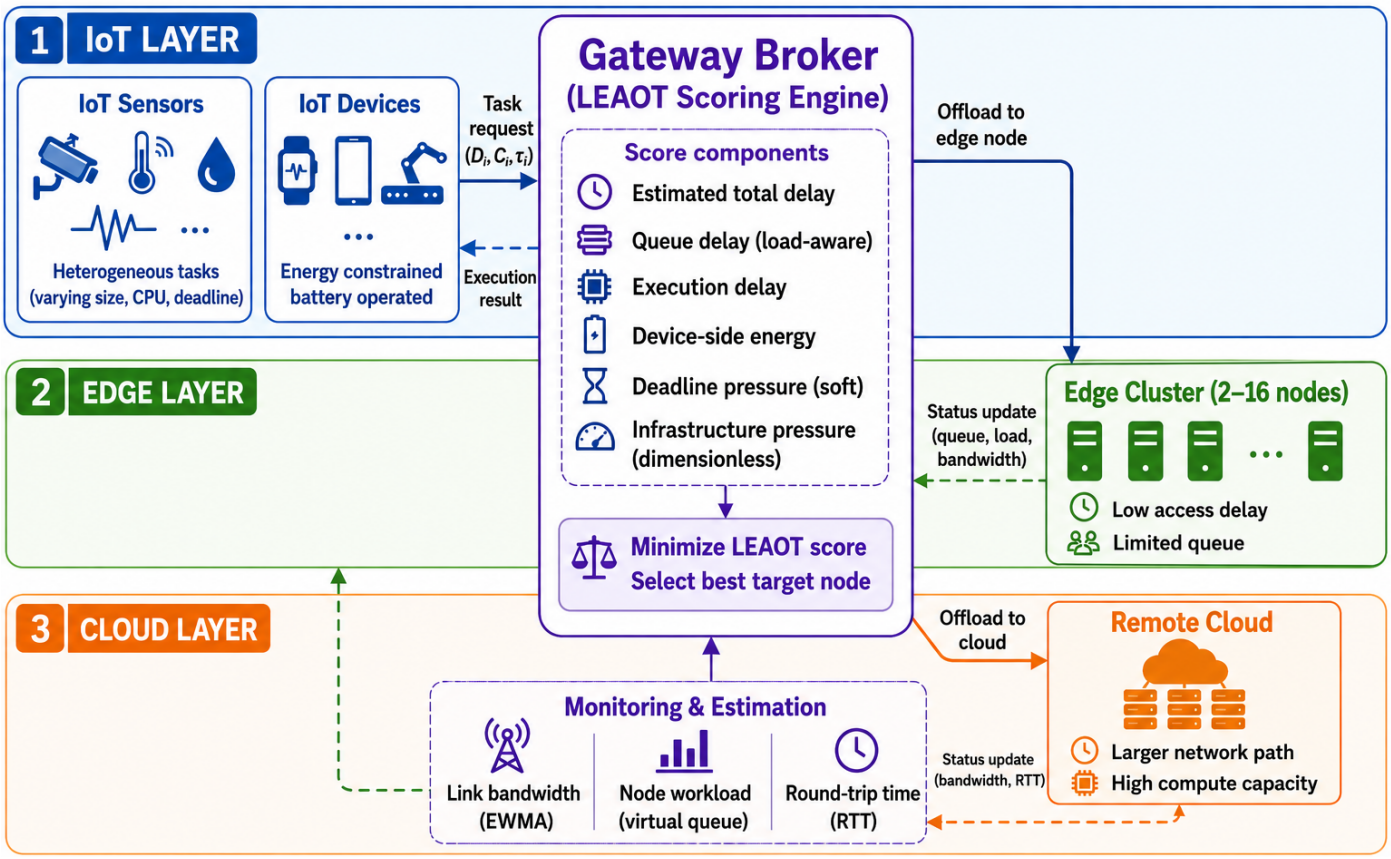

Fig. 1. IoT edge–cloud architecture used for the LEAOT evaluation. The edge cluster is scalable rather than fixed to two or three nodes.

workload (i.e., estimated using a virtual queue), and Round Trip Time (RTT). At each decision interval, the proposed *LEAOT* scoring engine leverages dynamic measurements to intelligently rank each execution target by jointly considering its estimated total execution delay, queue delay, execution time, device-side energy usage, and the combined effects of deadline and infrastructure pressure. Each task is then offloaded to the edge node/cloud with the lowest aggregated *LEAOT* score. The Edge Layer comprises a configurable cluster of 2–16 edge nodes that serve as compute resources, offering lower-latency execution for time-sensitive tasks. All edge nodes report current workload and network metrics to the Gateway Broker. While the Cloud Layer provides significantly more computation resources for heavy workloads, the increased number of network hops introduces much greater communication latency between the cloud and IoT devices. Cloud nodes also share feedback on their available bandwidth and RTT to the Gateway Broker for adaptive scheduling. Once completed, the chosen resource pushes the computation results back to the IoT device through the Gateway Broker.

Let $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$ be the arriving task stream and let $\mathcal{J} = \mathcal{E} \cup \{c\}$ be the candidate processing set, where $\mathcal{E}$ is the set of edge nodes and c is the cloud. Each task is represented as

$$T_i = \{a_i, D_i, C_i, \tau_i\}, \tag{1}$$

where, a_i is the arrival time, D_i is the input data size in MB, C_i is the required million CPU cycles, and τ_i is the deadline. Node j has aggregate processing rate F_j and unfinished workload $W_j(t)$ in million CPU cycles.

Associated with each task i is a relative deadline τ_i, which is the allowed time for completion starting from the moment it arrives at the Gateway Broker. Hence, task i meets its timing constraint if its expected total completion time meets $T_{ij}^{tot} \leq \tau_i$, where T_{ij}^{tot} is the total elapsed time from arrival to completion. In this study, τ_i will be considered a relative deadline, not an absolute time stamp.

B. Queue and Link Estimation

The queue model is a work-conserving FCFS aggregate-service approximation. Multi-core edge servers can be represented through the aggregate rate F_j. Before processing a new arrival, the unfinished workload is corrected as:

$$W_j(a_i) = \max\left(0, W_j(a_{i-1}) - F_j(a_i - a_{i-1})\right). \tag{2}$$

This correction prevents the virtual queue from drifting indefinitely when tasks finish between two arrivals. For candidate node j, the estimated queueing delay and execution delay are:

$$T_{ij}^{q} = \frac{W_j(a_i)}{F_j}, \qquad T_{ij}^{ex} = \frac{C_i}{F_j}. \tag{3}$$

Wireless bandwidth is time varying. *LEAOT* therefore does not treat B_{ij} as a permanent constant. The gateway maintains an Exponentially Weighted Moving Average (EWMA):

$$\widehat{B}_{ij}(t) = \xi \widehat{B}_{ij}(t-1) + (1-\xi) B_{ij}^{obs}(t), \tag{4}$$

where, B_{ij}^{obs} is the latest observed throughput and $0 < \xi < 1$ controls smoothing. The transmission delay is:

$$T_{ij}^{tx} = \frac{8D_i}{\widehat{B}_{ij}} + L_{ij}, \tag{5}$$

where, L_{ij} is the access delay and the factor 8 converts MB/Mbps to seconds. The estimated completion delay is:

$$T_{ij}^{tot} = T_{ij}^{tx} + T_{ij}^{q} + T_{ij}^{ex}. \tag{6}$$

C. Energy, Deadline, and Infrastructure Pressure

The energy term in this study is device-side energy. Server-side energy is important for data-center sustainability, but it is not the primary energy metric here. The device energy for assigning task i to node j is:

$$E_{ij} = P_i^{tx} T_{ij}^{tx} + P_i^{w} T_{ij}^{q}, \tag{7}$$

where, P_i^{tx} is transmit power and P_i^{w} is device waiting power. In particular, only the communication energy expenditure of the IoT device is modeled in the equation. Specifically, the first term models the transmission energy needed to upload the task onto the chosen execution node. The second term accounts for the idle energy expenditure of the device while waiting for execution to commence. Energy expenditure of the execution itself is omitted at either the edge or cloud as it does not impact the energy consumption of the IoT device (e.g., it is spent on the remote compute infrastructure instead). Thus, execution time impacts offloading decisions via the latency model, while device energy expenditure is captured by communications energy expenditure terms alone. The expected post-assignment load is:

$$\rho_{ij}^{+} = \min\left(1, \frac{W_j(a_i) + C_i}{F_j H}\right), \tag{8}$$

where, H is the scheduling horizon. Deadline pressure is modeled as a soft penalty rather than only a binary violation:

$$P_{ij}^{d} = \max\left(0, \frac{T_{ij}^{tot} - \tau_i}{\tau_i}\right). \tag{9}$$

Finally, the infrastructure-pressure index is:

$$I_{ij} = \eta_j \frac{C_i}{C_{max}} + \zeta_j \frac{D_i}{D_{max}}. \tag{10}$$

where, η_j and ζ_j are deterministic node-type indicator variables for distinguishing edge and cloud infrastructure in the infrastructure-pressure model. In particular, $\eta_j = 1$ and $\zeta_j = 0$ for all edge nodes, and $\eta_j = 0$ and $\zeta_j = 1$ for the cloud node. Through this binary encoding, the infrastructure-pressure index effectively manages the relative use of edge versus cloud resources without introducing additional variables to optimize over. In other words, this is not a currency cost. It is a normalized pressure term for computing and data movement. Cloud values of η_j and ζ_j are higher to represent remote resource and backhaul use, while edge load is already captured by ρ_{ij}^{+}.

For each task, each raw component x_{ij} is normalized over the candidate set:

$$\widetilde{x}_{ij} = \frac{x_{ij}}{\epsilon + \max_{k \in \mathcal{J}} x_{ik}}. \tag{11}$$

The *LEAOT* decision score is:

$$S_{ij} = \alpha \widetilde{T}_{ij}^{tot} + \beta \widetilde{E}_{ij} + \lambda \rho_{ij}^{+} + \delta \widetilde{P}_{ij}^{d} + \gamma \widetilde{I}_{ij}. \tag{12}$$

These weighting coefficients represent the relative significance of latency, energy efficiency, workload balancing, deadline satisfaction, and infrastructure utilization. Latency was weighted highest because most interactive IoT applications are delay-sensitive. Energy is generally considered the second-most important objective function for battery-powered devices. Queue load and deadline pressure are weighted equally as intermediate preferences for balanced resource utilization and on-time completion. The infrastructure-pressure term carries the lowest weight because it is intended only as a soft preference that penalizes heavy cloud use. Specifically, these values were chosen to give live application performance a significantly higher weight than an equitable distribution of load. The values here were based on initial sensitivity testing. The weighting coefficients are determined after doing several runs to figure out reasonable weights that satisfy the condition $\alpha + \beta + \lambda + \delta + \gamma = 1$.

Rather than searching for a globally optimal set of parameters, a Pareto trade-off between task finish time, energy used for communication, meeting deadlines, and network utilization was pursued. After some trial and error, the weights $\alpha = 0.34$, $\beta = 0.24$, $\lambda = 0.18$, $\delta = 0.18$, and $\gamma = 0.06$ were decided because they produced good results for all desired metrics. Additional justification can be found in Section IV-E through parameter sensitivity analysis.

The selected processing node is:

$$j^{*} = \arg\min_{j \in \mathcal{J}} S_{ij}. \tag{13}$$

While the ultimate decision is based on a normalized weighted score, it is important to emphasize that the presented *LEAOT* model is not merely a weighted-sum heuristic. The key contribution of this work is the fusion of complementary online decision heuristics within a single gateway-level scheduling model. In detail, virtual workload tracking is augmented to account for stragglers (i.e., stale queue estimates), use EWMA-based bandwidth predictions to react to time-varying wireless conditions, model soft deadline-pressures that account for different degrees of deadlines instead of imminent deadline

misses, and include a dimensionless infrastructure-pressure metric to penalize overly aggressive cloud-server usage without the need to hard-code provider-specific prices. These heuristics provide a holistic, explainable, low-complexity online scheduler that constantly adapts to changes in edge–cloud conditions without iterative optimization or training.

Algorithm 1 has computational complexity $O(NM)$, where N is the number of arriving tasks and M is the number of candidate processing nodes. Since each task is compared against every processing node once, complexity increases linearly with workload size and infrastructure scale. *LEAOT* iterates through the candidate node set only once, and performs no iterative optimization, reinforcement learning, or repeated model updating. For these reasons, *LEAOT* is well-suited for online gateway deployment, where scheduling decisions must be produced quickly.

Algorithm 1 LEAOT online latency–energy offloading

Require: Task stream $\mathcal{T}$, candidate nodes $\mathcal{J}$, weights $\alpha, \beta, \lambda, \delta, \gamma$, smoothing factor ξ
Ensure: The selected processing node for each task
Initialize $W_j(0)$ and $\widehat{B}_{ij}(0)$ for each node
for each task $T_i = \{a_i, D_i, C_i, \tau_i\}$ in arrival order **do**
  for each candidate node $j \in \mathcal{J}$ **do**
    Correct virtual workload using (2)
    Estimate T_{ij}^{tx}, T_{ij}^{q}, T_{ij}^{ex}, and T_{ij}^{tot} using (3)–(6)
    Compute E_{ij}, ρ_{ij}^{+}, P_{ij}^{d}, and I_{ij} using (7)–(10)
  end for
  Normalize candidate components using (11)
  Compute S_{ij} using (12)
  Select $j^* = \arg\min_{j \in \mathcal{J}} S_{ij}$
  Assign T_i to j^* and update $W_{j^*} \leftarrow W_{j^*} + C_i$
  Update the link estimate with new observations using (4)
end for

IV. Performance Evaluation

A. Simulation Environment and Workload Generation

Unlike machine learning-empowered studies, the proposed *LEAOT* model was not evaluated on a real-world or private dataset. Instead, the experiments were conducted using a reproducible simulator built on top of a mathematical edge–cloud task-offloading model. Synthetic IoT workloads and network conditions representative of realistic edge computing use cases are used in simulation tests.

Simulated workloads consist of heterogeneous IoT devices that offload computational requests to multiple edge servers (proxied) and to one remote cloud server. Edge servers have different computing capabilities, communication latencies, bandwidth allowances, and resource granularities, and the cloud offers orders-of-magnitude higher compute resources at the cost of higher latency. Bandwidth conditions can change throughout the program execution, mimicking real wireless communications environments. Moreover, bandwidth estimates are assumed to be maintained online using an Exponentially Weighted Moving Average (EWMA) estimator.

In the simulation environment, the simulator generates heterogeneous IoT tasks with distinct workload patterns that model latency-sensitive, medium-complexity, and computation-intensive IoT applications. Tasks are described by four parameters, including arrival time, input data size, computational workload, and execution deadline. IoT tasks arrive according to a Poisson arrival process. All other task parameters are sampled from preset statistical distributions. In particular, tasks arrive at the system according to a Poisson process, with interarrival times exponentially distributed. Tasks' input data sizes, workload levels, execution deadlines, transmission power, and waiting power are sampled independently from preset uniform distributions that mimic heterogeneous workload requests from different IoT applications. Three workload levels are considered for each IoT task to mimic lightweight, medium- complexity, and computation-intensive tasks. Moreover, stochastic multiplicative noise is applied to dynamically perturb the available bandwidth during each simulation period. Also, channel variations are emulated by continuously tracking the available bandwidth using an EWMA estimator.

Experiments are conducted using a randomly generated workload of 1,000 IoT tasks and under edge–cloud infrastructures with 2 to 16 edge nodes and a remote cloud server. Each experiment uses 10 independent random seeds to generate stochastic variations in workload requests and network dynamics. Performance metrics are reported as averages and accompanied by the 95% confidence intervals.

The developed *LEAOT* algorithm is evaluated against five representative baselines, including Cloud-only, QAE-baseline, Threshold-based offloading, DRC-baseline, and DPP-baseline. Each algorithm is evaluated using multiple metrics, including average task latency, device energy consumption, deadline violation rate, cloud utilization, and scalability with respect to the edge node count.

Since all workloads are synthetically generated within the simulator according to predefined statistical distributions, there is no dataset. However, simulating the workload within the environment allows us to create controlled, repeatable, and fair benchmarks between offloading strategies while realistically capturing heterogeneous IoT workloads and dynamic edge–cloud settings.

All offloading schemes were compared under the same simulation settings to ensure a fair comparison. Independent random seeds were used for each scheme run. Reported figures represent average performance across 10 runs, with a 95% confidence interval. Full simulation code also produces the numerical results and publication-quality figures from simulation output, allowing easy replication or future comparisons.

B. Evaluation Setup

The proposed simulator is implemented in Python 3 using NumPy for numerical routines, Pandas for statistics aggregation and result storage, and Matplotlib for plotting. Conceptually, the simulator runs in discrete events, where new task arrivals, queue progression, bandwidth changes, and scheduling decisions are applied in timestamp order. The simulator automatically produces all figures and tables from stored simulation results. All experimental results can be reproduced.

A discrete-event evaluation model was used to generate task arrivals, update node queues, emulate stochastic band-

TABLE II. REFERENCE SIMULATION PARAMETERS

Parameter	Value
Tasks in reference setting	1000
Task input size D_i	0.15–5.0 MB, mixed workload
CPU demand C_i	180–3600 Mcycles
Task deadline τ_i	0.35–2.50 s
Edge nodes in reference setting	8
Scalability test	2, 4, 8, and 16 edge nodes
Edge CPU rate	2600–5200 Mcycles/s
Cloud CPU rate	14500 Mcycles/s
Edge bandwidth	28–100 Mbps
Cloud bandwidth	18–42 Mbps
Edge access latency	6–28 ms
Cloud access latency	80–145 ms
Transmit and waiting power	0.45–0.80 W; 0.055–0.095 W
Edge-node indicator (η_j)	1 (edge), 0 (cloud)
Cloud-node indicator (ζ_j)	0 (edge), 1 (cloud)
LEAOT weights	$\alpha = 0.34$, $\beta = 0.24$, $\lambda = 0.18$, $\delta = 0.18$, $\gamma = 0.06$

width variation, and apply each offloading policy under identical traffic and node conditions. The evaluation intentionally avoids the parameter ranges in Table II. All results reported in this section are averaged over ten independent trials, and error bars show 95% confidence intervals.

LEAOT is compared with five baselines. Cloud-only sends every task to the cloud. Threshold selection uses the nearest acceptable edge node if its estimated load and deadline are acceptable; otherwise, it uses the cloud. The Queue-Aware Edge–cloud (QAE) baseline selects the least-loaded edge node but forwards the task to the cloud when the predicted edge delay or utilization is high. It first selects:

$$j_e^* = \arg\min_{j \in \mathcal{E}} \rho_{ij}^+, \tag{14}$$

and then applies:

$$x_i = \begin{cases} j_e^*, & T_{ij_e^*}^{tot} \le T_{th},\ \rho_{ij_e^*}^+ \le \rho_{th}, \\ c, & \text{otherwise}, \end{cases} \tag{15}$$

where, $T_{th} = 1.05\tau_i$ and $\rho_{th} = 0.76$. The Delay-Resource-Cost (DRC) baseline follows the objective family used in recent edge–cloud offloading studies by combining normalized delay and infrastructure pressure, but without the *LEAOT* energy and soft deadline terms [11], [12]. The Drift-Plus-Penalty (DPP) baseline is a Lyapunov-inspired lightweight heuristic that combines delay, energy, and queue load.

The threshold baseline uses a static decision threshold for all the experiments and does not tune the decision boundary based on current network load, queue size, or workload variation. Specifically, this setup simulates a typical static heuristic and provides a baseline to compare against adaptive offloading strategies.

C. Main Results with Confidence Intervals

Table III presents the aggregated results of all policies when operating under a reference workload of 1000 IoT tasks submitted to eight edge nodes with ten independent random seeds each. As expected, Cloud-only yields the longest latency 602.8 ms, the highest energy usage 0.274 J/task, and the worst deadline violation rate 11.43%, confirming that offloading all tasks exclusively to remote cloud data centers is highly inefficient. Threshold also fares poorly because its predefined offloading threshold does not account for workload fluctuations and is therefore outperformed by Cloud-only in both latency 644.8 ms and violation ratio 11.81%. Compared against Cloud-only, QAE-baseline improves latency and energy by a large margin due to cloud fallback while reducing violation ratio to 1.40%; however, since QAE does not explicitly account for energy or infrastructure, it fares worse than both DRC and *LEAOT*. DRC-baseline obtains one of the lowest energy figures of 0.117 J/task and offloads the least amount of tasks to the cloud, 0.03%, but suffers from a higher violation ratio of 0.89% compared to *LEAOT*. DPP achieves the lowest average latency, 387.2 ms, but incurs this benefit at the cost of using the cloud 6.73% more often than DRC, resulting in worse energy efficiency of 0.129 J/task. Meanwhile, *LEAOT* matches DRC's energy consumption of 0.117 J/task, achieves the lowest deadline violation ratio among all competing methods at 0.69%, and incurs only a marginally higher latency, around 2.1% greater than DPP, while using the cloud 5.25 times less than DPP. Although there is room for improvement in latency, *LEAOT* provides an order-of-magnitude better trade-off between energy-delay-cost and infraction than its baselines. This showcases that the proposed multi-objective cost function achieves its design purpose of jointly considering execution delay, energy cost, and infrastructure.

The Threshold policy minimizes communication delay because it assigns small tasks to local edge nodes. However, its decisions are made offline and do not consider the current workload or queue lengths at edge nodes. During high-load conditions, several devices may choose the same set of edge servers, which can cause queuing delay to increase substantially, offsetting the benefit of lower communication delay. By comparison, the Cloud-only policy sends tasks directly to the cloud. While this incurs higher communication delay, there is no queuing delay at the edge servers. For the workload used in the evaluation, the queuing delay incurred by the Threshold policy is slightly higher than the communication overhead to the cloud, and as a result, its overall latency and deadline miss rate are slightly higher than the Cloud-only policy.

Fig. 2 and 3 illustrate the average latency and device energy consumption of each evaluated offloading policy with 95% confidence intervals over ten runs. Both Cloud-only and Threshold incur the highest latency (>600 ms), which validates the intuition that excessive cloud executions or non-adaptive threshold-based decision-making will lead to poor task completion time. Thanks to the adaptability of edge-assisted ap-

TABLE III. REFERENCE RESULTS FOR 1000 TASKS, 8 EDGE NODES, AND 10 SEEDS

Policy	Latency ms	Energy J/task	Violation %	Cloud %
Cloud-only	602.8 ± 23.6	0.274 ± 0.013	11.43 ± 1.67	100.00
QAE-baseline	417.6 ± 22.7	0.146 ± 0.014	1.40 ± 0.41	1.96
Threshold	644.8 ± 16.7	0.223 ± 0.015	11.81 ± 1.61	44.46
DRC-baseline	398.8 ± 20.9	0.117 ± 0.007	0.89 ± 0.31	0.03
DPP-baseline	387.2 ± 16.2	0.129 ± 0.010	0.72 ± 0.21	6.73
LEAOT	395.4 ± 18.3	0.117 ± 0.007	0.69 ± 0.24	1.48

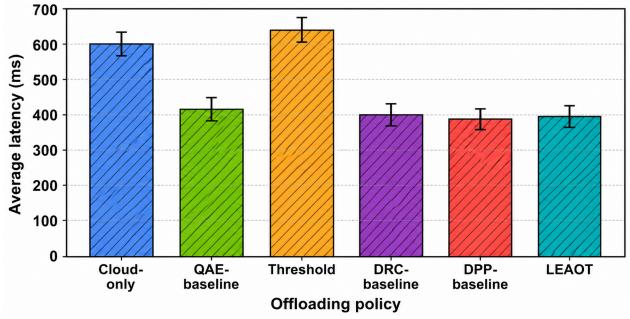

Fig. 2. Average latency with 95% confidence intervals over ten independent seeds.

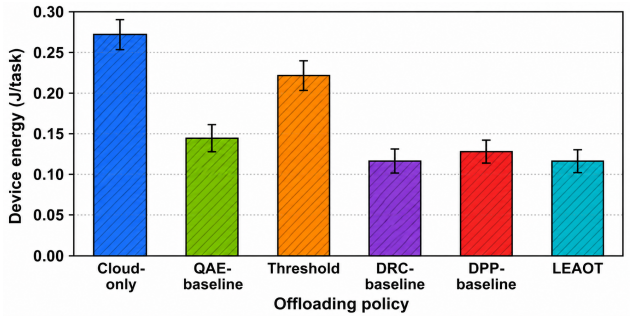

Fig. 3. Device-side energy with 95% confidence intervals. LEAOT keeps energy close to the lowest-energy baseline while improving deadline satisfaction.

proaches, latency is reduced across all smart devices, with DPP-baseline achieving the lowest latency at 387.2 ms on average, followed by *LEAOT* at 395.4 ms and DRC-baseline at 398.8 ms. As for energy consumption, Cloud-only has the highest energy usage at 0.274 J/task, while both DRC-baseline and *LEAOT* have the lowest average energy consumption at 0.117 J/task, highlighting the benefit of accounting for energy costs during edge scheduling. While *LEAOT* does not achieve the lowest latency, it can be seen that *LEAOT*'s latency is within 2% of the best approach while achieving the lowest deadline violation rate (i.e., Table III) and one of the lowest energy costs. In addition, the confidence intervals are relatively tight, indicating that the performance difference is consistent across different random seeds.

D. Load and Scalability Analysis

Fig. 4 plots the deadline violation ratio as the number of IoT tasks scales up from 400 to 1300. It is observed that the proposed *LEAOT* achieves the lowest deadline violation

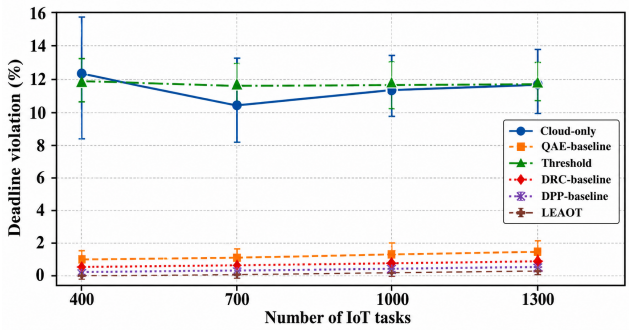

Fig. 4. Deadline violation under increasing IoT task load. Results are averaged over ten seeds.

across all workload levels, with only a slight degradation under heavier load. This shows that jointly considering latency, energy, and infrastructure pressure allows the scheduler to be robust to increasing traffic demand. DPP-baseline also enjoys low violation ratios since load-awareness is built into its decisions, but incurs higher violations overall compared to *LEAOT*. DRC-baseline aims to directly minimize energy costs in resource utilization and does not explicitly penalize deadline pressure, thus exhibiting slightly higher violation ratios as the workload increases. The QAE baseline shows a gradual increase in violations because it lacks explicit awareness of infrastructure pressure in its decision-making process. The threshold policy exhibits the worst performance among edge-based schemes, as its heuristic decision threshold cannot adapt to heterogeneous task arrivals and varying resource conditions. A cloud-only policy suffers from a high violation ratio regardless of workload, since remote cloud execution incurs significant communication delays. The relatively narrow confidence interval further demonstrates the stability of these trends across 10 random seeds.

Scalability performance is demonstrated by holding the workload constant and gradually increasing the number of edge nodes from 2 to 16. From Fig. 5, it is observed that all edge-assisted offloading methods improve with increasing amounts of edge resources, exhibiting smaller average latencies. As expected, *LEAOT* shows clear gains from scaling since more edge nodes allow choosing from a candidate set with a higher probability of selecting lightly-loaded nodes and avoiding overloaded ones. Average latency improves from around 562 ms with two edge nodes to around 364 ms with sixteen edge nodes. While DPP-baseline exhibits the smallest latency, it offloads more heavily to the cloud than *LEAOT* does. From Table IV, it can be seen that *LEAOT* achieves similar latency while reducing cloud dependency and edge congestion.

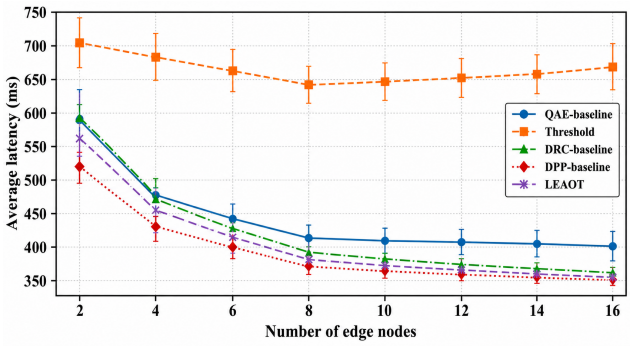

Fig. 5. Scalability with different edge-node counts. The topology is varied from 2 to 16 edge nodes, and error bars show 95% confidence intervals.

Threshold does not exhibit meaningful gains as the number of edge nodes increases, since its rule cannot leverage the added resource heterogeneity. Both QAE and DRC baselines see improvements with larger edge-node deployments, but are generally less capable of delivering competitive results than *LEAOT* does across latency, energy, and cloud usage. The confidence intervals are relatively small, suggesting that the increase in edge nodes has similar effects across the ten random seeds. These results suggest that *LEAOT* can scale with the number of edge nodes available, which aligns with its theoretical $O(NM)$ computation complexity.

Table IV presents numerical results to further show that *LEAOT* scales with different number of edge nodes. Increasing the number of available edge nodes from 2 to 16 consistently decreases the average latency from 562.7 ms to 363.7 ms, energy J consumption from 0.188 to 0.107, deadline violation rate from 6.36% to 0.44%, and cloud offloading ratio from 27.26% to 0.56%. Such improvement achieved by intermediate configurations (i.e., 4, 6, 8, 10, 12, and 14 edge nodes) is smooth and monotonic, indicating that more edge resources enable the broker to select nodes closer to the request, resulting in lower communication delay and a lighter workload. As a result, fewer tasks are forwarded to the cloud for execution, which reduces both transmission overhead and deadline violations. Notice that *LEAOT* scales naturally with existing infrastructure without retraining as the number of edge nodes varies; it always evaluates the advertised candidate set and selects the best-fit execution node.

E. Effect of the Infrastructure-Pressure Weight

The sensitivity analysis is limited to γ, since γ is the only parameter that directly controls the trade-off between optimizing individual task cost and equalizing edge-cloud resource utilization. α, β, λ, and δ are kept at their empirically chosen values in order to remove confounding factors from other parameter variations. Recall that the infrastructure-pressure index computed in (10) represents resource usage/congestion, and not a cost in dollars. Fig. 6 shows how *LEAOT*'s behavior varies with infrastructure-pressure weight γ; other parameters are held constant for all experiments here. By design, $\gamma = 0$ causes *LEAOT* to focus exclusively on latency, aggressively offloading computation to higher-capacity remote servers. This policy achieves the lowest possible normalized latency, but results in the highest infrastructure-pressure index. Raising γ causes

TABLE IV. SCALABILITY OF LEAOT WITH EDGE-NODE COUNT

Edges	Latency ms	Energy J	Violation %	Cloud %
2	562.7 ± 30.9	0.188 ± 0.014	6.36 ± 2.24	27.26
4	453.1 ± 30.8	0.140 ± 0.012	2.15 ± 1.35	5.31
6	423.0 ± 25.2	0.127 ± 0.010	1.34 ± 0.82	3.42
8	392.9 ± 19.6	0.115 ± 0.007	0.73 ± 0.30	1.47
10	385.5 ± 17.4	0.112 ± 0.006	0.62 ± 0.27	1.18
12	378.2 ± 15.6	0.110 ± 0.006	0.54 ± 0.24	0.89
14	371.0 ± 13.8	0.108 ± 0.005	0.48 ± 0.22	0.71
16	363.7 ± 12.1	0.107 ± 0.005	0.44 ± 0.20	0.56

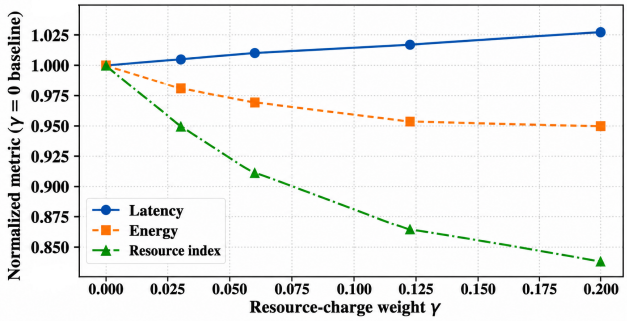

Fig. 6. Sensitivity of latency, energy, and infrastructure pressure to the resource-pressure weight γ. Metrics are normalized to the $\gamma = 0$ setting.

the broker to more strictly limit the use of remote resources; infrastructure pressure drops dramatically as remote executions are curtailed, and energy consumption also decreases (but to a lesser extent). This effect occurs with only a small uptick in normalized latency, illustrating that the proposed objective seamlessly tunes the latency-pressure trade-off rather than sharply altering scheduler decisions. Based on these results, $\gamma \in [0.06, 0.12]$ achieves near-optimal latency while reducing pressure on network/cloud resources, whereas larger values of γ can be used to aggressively minimize infrastructure usage at the cost of slightly higher latency. Thus, γ can be used as a knob for operators to tune their deployment of *LEAOT* without changing the underlying scheduler.

Table V explores the sole effect of infrastructure- pressure weight γ on the proposed *LEAOT* scheduler. Clearly, by progressively giving more weight to less congested infrastructure as γ increases from 0.00 to 0.20, the infrastructure-pressure index decreases from 0.192 ± 0.016 to 0.161 ± 0.008, while cloud utilization drops significantly from 3.61% to 0.10%. In other words, more tasks can be fitted into the edge infrastructure as the traffic load is better balanced across infrastructure nodes. Meanwhile, the cost incurred by this better balance of infrastructure pressure is negligible, as average latency only grows by 10.6 ms, about 2.7% overall, while average

energy consumption slightly decreases from 0.119 ± 0.008 to 0.113 ± 0.006. This result empirically shows that γ is an effective knob for controlling traffic load across edge infrastructure, enabling more tasks to be run locally and reducing reliance on the cloud, while keeping average performance in both latency and energy nearly identical. As a trade-off, operating at γ=0.12 achieves a comparable reduction in infrastructure pressure and cloud usage, with only mild growth in execution latency.

TABLE V. ISOLATED SENSITIVITY OF THE INFRASTRUCTURE-PRESSURE WEIGHT

γ	**Latency ms**	**Energy J**	**Pressure**	**Cloud %**
0.00	389.4 ± 18.3	0.119 ± 0.008	0.192 ± 0.016	3.61
0.03	391.4 ± 18.9	0.117 ± 0.008	0.183 ± 0.014	2.49
0.06	394.0 ± 19.7	0.116 ± 0.007	0.175 ± 0.012	1.54
0.12	397.0 ± 20.4	0.114 ± 0.006	0.166 ± 0.010	0.63
0.20	400.0 ± 21.9	0.113 ± 0.006	0.161 ± 0.008	0.10

V. DISCUSSION AND LIMITATIONS

From the experimental results, *LEAOT* achieves a favorable trade-off among all objectives (i.e., latency, device energy consumption, deadline miss rate, and processing resource utilization). The proposed framework jointly models multiple competing objectives to make scheduling decisions, rather than optimizing a single metric. Therefore, the approach can yield more reliable scheduling decisions than the baselines across various conditions in heterogeneous edge–cloud systems. As shown in the experiments, *LEAOT* reduces overall missed deadlines and energy consumption on devices while offering comparable latency to optimization- and learning-based baselines. At the same time, it does not sacrifice excessive amounts of one objective when improving another. Such characteristics are especially important for real-time IoT applications that are sensitive to specific objectives. For example, aggressive minimization of latency could easily hurt deadline satisfaction.

LEAOT can be deployed in practice online at the gateway or edge-broker level. Unlike the iterative computations required by optimization-based methods and the periodic training and retraining procedures required by learning-based methods, *LEAOT* only needs the current state of the system for score-based, lightweight scheduling. Therefore, its time complexity grows linearly with the number of candidate nodes for scheduling, enabling real-time scheduling decisions in resource-constrained edge–cloud environments. In addition, the reported performance numbers have yet to be tested with real IoT workloads, standard open-source edge computing benchmark suites, or even real edge computing systems themselves. Thus, all reported latency, energy usage, and deadline hit rates should be considered calibrated comparative results and not necessarily indicative of deployed performance. *LEAOT* will be tested with realistic workload traces and industry-standard benchmark suites such as EdgeCloudSim in the future. Further experiments with realistic communication characteristics and edge testbeds consisting of diverse IoT hardware and edge servers are planned.

While highlighting these benefits, several limitations must also be noted. First, the queueing theoretic model adopts an FCFS aggregate-service approximation to predict average waiting time rather than capturing fine-grained processor scheduling policies (e.g., priority queues, preemption, heterogeneous cores with multiple concurrent executions, etc.). Second, the proposed energy model captures end-device energy consumption associated with communication and waiting, while neglecting energy consumed by servers. Third, while bandwidth variability is captured by stochastic variations and continuously tracked by the EWMA predictor, practical wireless networks may exhibit richer temporal correlations that demand more elaborate predictors. Fourth, the weighting coefficients of the *LEAOT* score were arbitrarily chosen to present a normalized trade-off between objectives. While sensitivity analysis shows that the proposed model is not highly sensitive to small fluctuations in weighting parameters, the automatic adaptation of weights is an interesting topic for future work.

Moreover, another limitation of the evaluation is that the task arrivals are modeled using a Poisson process. While Poisson traffic is commonly used as a benchmark for performance evaluation of task scheduling and resource allocation algorithms, many real-world IoT systems experience bursty, periodic, correlated, or heavy-tailed traffic patterns. Traffic behaviors that are not well-modeled by Poisson processes may exhibit different queueing behavior and affect waiting times and decisions to offload tasks. Thus, although the results show the performance of *LEAOT* under certain settings, more practical traffic patterns need to be considered to evaluate its robustness across various scenarios.

Finally, the experimental evaluation of the proposed model is entirely simulation-based, relying on the synthetic generation of heterogeneous IoT workloads. Although this methodology allows us to conduct controlled, reproducible, and statistically valid performance evaluation, validating the proposed model on public edge-computing benchmarking suites, real IoT traces, and/or physical edge deployments would be valuable. Adaptive weight optimization, mobility-aware scheduling, and joint learning with reinforcement or federated learning algorithms could also enhance the model's versatility for large-scale edge intelligence.

VI. CONCLUSION AND FUTURE WORK

This study presented *LEAOT*, a load-, deadline-, latency-, and energy-aware task offloading method for IoT Edge–Cloud Systems. *LEAOT* introduces a unified online task-offloading model that seamlessly integrates virtual workload calibration, online bandwidth tracking, queue-aware latency prediction, device-side energy estimation, soft-deadline pressure calculation, and infrastructure pressure perception into a single explainable decision-making model. Distinguished from existing optimization- and learning-based solutions, it accomplishes this without iterative solution methods or model training, yet achieves comparable performance across various edge–cloud scenarios. The method also incorporates EWMA link estimation and virtual workload correction so that bandwidth and queue states are not treated as static constants. Simulation

results show that *LEAOT* achieves low device energy and the lowest deadline violation in the reference setting while remaining competitive with DRC and DPP baselines. The node-scaling analysis further shows that *LEAOT* can operate across edge topologies from 2 to 16 nodes without retraining. Future work will extend the method to real IoT traces, server-side energy modeling, mobility-aware multi-gateway orchestration, and security-aware task placement.

While the experimental results validate the efficacy of the proposed *LEAOT* model, several directions for future work are identified. First, the current evaluation focuses on a simulated testbed; deploying *LEAOT* on real-world edge platforms and large-scale IoT testbeds will help validate its practicality. Second, adaptive methods for learning optimal weights are planned to tune scoring coefficients on the fly based on network dynamics, application requirements, and workload features, rather than relying on handcrafted parameters. Third, advanced workload prediction and queue estimation techniques, such as predictive modeling or lightweight ML models, can be integrated to enhance decision-making accuracy in highly dynamic scenarios. Furthermore, extending *LEAOT* to account for task dependencies and user mobility patterns, support heterogeneous wireless interfaces (e.g., 5G or 6G), and scale to multi-tier edge computing environments will enable its adoption across broader IoT systems. Moreover, the performance of the proposed *LEAOT* system using realistic IoT traffic models will be evaluated, such as ON/OFF burst arrivals, synchronized periodic traffic, correlated traffic generation, and heavy-tailed interarrival times, to better understand its performance stability under various types of network traffic while still holding the offered load similar between the competitive offloading schemes. Finally, incorporating security, privacy, and energy consumption metrics on edge servers will be important for making holistic and sustainable offloading decisions in future IoT–Edge–Cloud systems. Furthermore, future work involves testing the proposed system with public edge computing benchmark suites, IoT workload traces, and real edge computing deployments to further validate scalability, robustness, and real-world applicability.

REFERENCES

[1] D. Witczak and S. Szymoniak, "Review of monitoring and control systems based on internet of things," *Applied Sciences*, vol. 14, no. 19, p. 8943, 2024.

[2] K. Micko, P. Papcun, and I. Zolotova, "Review of iot sensor systems used for monitoring the road infrastructure," *Sensors*, vol. 23, no. 9, p. 4469, 2023.

[3] S. Ki, G. Byun, K. Cho, and H. Bahn, "Co-optimizing cpu voltage, memory placement, and task offloading for energy-efficient mobile systems," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 9177–9192, 2023.

[4] S. Akter, D. V. A. Duong, D.-Y. Kim, and S. Yoon, "Task offloading and resource allocation in uav-aided emergency response operations via soft actor critic," *IEEE Access*, vol. 12, pp. 69 258–69 275, 2024.

[5] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das, "Edge-computing-driven internet of things: A survey," *ACM computing surveys*, vol. 55, no. 8, pp. 1–41, 2022.

[6] F. S. Abkenar, P. Ramezani, S. Iranmanesh, S. Murali, D. Chulerttiyawong, X. Wan, A. Jamalipour, and R. Raad, "A survey on mobility of edge computing networks in iot: State-of-the-art, architectures, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2329–2365, 2022.

[7] S. Douch, M. R. Abid, K. Zine-Dine, D. Bouzidi, and D. Benhaddou, "Edge computing technology enablers: A systematic lecture study," *IEEE Access*, vol. 10, pp. 69 264–69 302, 2022.

[8] G. I. Arcas, T. Cioara, I. Anghel, D. Lazea, and A. Hangan, "Edge offloading in smart grid," *Smart Cities*, vol. 7, no. 1, pp. 680–711, 2024.

[9] J. Almutairi and M. Aldossary, "A novel approach for iot tasks offloading in edge-cloud environments," *Journal of cloud computing*, vol. 10, no. 1, p. 28, 2021.

[10] S. Dong, J. Tang, K. Abbas, R. Hou, J. Kamruzzaman, L. Rutkowski, and R. Buyya, "Task offloading strategies for mobile edge computing: A survey," *Computer Networks*, vol. 254, p. 110791, 2024.

[11] L. Liu, H. Zhu, T. Wang, and M. Tang, "A fast and efficient task offloading approach in edge-cloud collaboration environment," *Electronics*, vol. 13, no. 2, p. 313, 2024.

[12] S. Li, W. Li, H. Liu, and W. Sun, "Tiered-pricing-based task offloading strategy in collaborative edge-cloud computing: A matching game approach," *IEEE Internet of Things Journal*, vol. 12, no. 11, pp. 15 114–15 129, 2025.

[13] A. Benaboura, R. Bechar, W. Kadri, T. D. Ho, Z. Pan, and S. Sahmoud, "Latency-aware and energy-efficient task offloading in iot and cloud systems with dqn learning," *Electronics*, vol. 14, no. 15, p. 3090, 2025.

[14] B. Bahrami, M. R. Khayyambashi, and S. Mirjalili, "Edge server placement problem in multi-access edge computing environment: models, techniques, and applications," *Cluster Computing*, vol. 26, no. 5, pp. 3237–3262, 2023.

[15] H. Sfaxi, I. Lahyani, S. Yangui, and M. Torjmen, "Latency-aware and proactive service placement for edge computing," *IEEE Transactions on Network and Service Management*, vol. 21, no. 4, pp. 4243–4254, 2024.

[16] A. A. Amer, I. E. Talkhan, R. Ahmed, and T. Ismail, "An optimized collaborative scheduling algorithm for prioritized tasks with shared resources in mobile-edge and cloud computing systems," *Mobile Networks and Applications*, vol. 27, no. 4, pp. 1444–1460, 2022.

[17] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[18] Kanupriya, I. Chana, and R. K. Goyal, "Computation offloading techniques in edge computing: A systematic review based on energy, qos and authentication," *Concurrency and Computation: Practice and Experience*, vol. 36, no. 13, p. e8050, 2024.

[19] H. A. Alharbi, M. Aldossary, J. Almutairi, and I. A. Elgendy, "Energy-aware and secure task offloading for multi-tier edge-cloud computing systems," *Sensors*, vol. 23, no. 6, p. 3254, 2023.

[20] Z. Du, C. Peng, T. Yoshinaga, and C. Wu, "A q-learning-based load balancing method for real-time task processing in edge-cloud networks," *Electronics*, vol. 12, no. 15, p. 3254, 2023.

[21] H. Li, X. Zhang, H. Li, X. Duan, and C. Xu, "Sla-based task offloading for energy consumption constrained workflows in fog computing," *Future Generation Computer Systems*, vol. 156, pp. 64–76, 2024.

[22] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya, "Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments," *Future Generation Computer Systems*, vol. 152, pp. 55–69, 2024.

[23] K. A. Bui and M. Yoo, "Interruption-aware computation offloading in the industrial internet of things," *Sensors*, vol. 25, no. 9, p. 2904, 2025.

[24] Y. Dai, J. Zhao, J. Zhang, Y. Zhang, and T. Jiang, "Federated deep reinforcement learning for task offloading in digital twin edge networks," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 3, pp. 2849–2863, 2024.

[25] X. Chen, J. Cao, R. Cao, Y. Sahni, M. Zhang, and Y. Ji, "Decentralized task offloading in collaborative edge computing: a digital twin assisted multi-agent reinforcement learning approach," *IEEE Transactions on Mobile Computing*, 2025.