Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Enhanced Lemurs Optimizer for Multi-Objective Task Scheduling in Heterogeneous Fog–Cloud Architectures

Nedal Ababneh

Nedal.Ababneh@ukf.ac.ae

University of Khorfakkan

Ammar Kamal Abasi

Abu Dhabi Polytechnic

Yasir Hamid

Abu Dhabi Polytechnic

Artur Koci

Abu Dhabi Polytechnic

Research Article

Keywords: Fog–Cloud Scheduling, Multi-objective Optimiza- tion, Enhanced Lemurs Optimizer (ELO), Energy-efficient Com- puting, Task Allocation Algorithms

Posted Date: April 1st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8658789/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Enhanced Lemurs Optimizer for Multi-Objective Task Scheduling in Heterogeneous Fog–Cloud Architectures

Nedal Ababneh*, Ammar Kamal Abasi†, Yasir Hamid†, and Artur Koci†

*College of Computing and Intelligent Systems, University of Khorfakkan, Sharjah, UAE
Emails: {Nedal.Ababneh}@ukf.ac.ae
†Information Security Engineering Technology, Abu Dhabi Polytechnic, MBZ, Abu Dhabi, 111499, UAE
Emails: {ammar.abasi, Yasir.Hamid, Artur.Koci}@actvet.gov.ae
Corresponding author: Nedal Ababneh (Nedal.Ababneh@ukf.ac.ae)

***Abstract*—Task scheduling plays a central role in Fog–Cloud computing systems, where heterogeneous resources and latency-sensitive workloads require intelligent distribution of tasks to ensure efficient operation. However, achieving optimal scheduling is challenging because the problem is NP-hard and involves competing objectives—minimizing makespan, reducing energy consumption, and lowering processing cost. Existing heuristic and metaheuristic methods often struggle to maintain a proper balance between exploration and exploitation, resulting in premature convergence or suboptimal solutions. This paper proposes an Enhanced Lemurs Optimizer (ELO) that integrates a problem-aware heuristic initialization with the adaptive search dynamics of the original Lemurs Optimizer (LO). The heuristic phase generates high-quality starting schedules by considering execution time, power efficiency, and cost trade-offs, while the LO refinement phase stabilizes convergence through controlled exploration and exploitation. Extensive simulations across workloads of 100–400 tasks demonstrate that ELO consistently outperforms benchmark algorithms. Compared to GA, ELO achieves up to 54% lower makespan, up to 52% lower energy consumption, and up to 11% lower processing cost, while also exhibiting substantially more stable convergence behavior. Wilcoxon signed-rank tests confirm that these improvements are statistically significant across all experimental settings. These results highlight the potential of the ELO as an effective and scalable scheduling framework for real-world Fog–Cloud systems.**

***Index Terms*—Fog–Cloud Scheduling, Multi-objective Optimization, Enhanced Lemurs Optimizer (ELO), Energy-efficient Computing, Task Allocation Algorithms**

I. Introduction

The proliferation of the Internet of Things (IoT), smart cities, Industry 4.0 systems, and cyber–physical infrastructures has generated unprecedented volumes of data requiring real-time or near–real-time processing [1]. Recent forecasts estimate the number of connected IoT devices could exceed 39 billion by 2030, potentially generating tens of zettabytes of data annually—creating strong demand for edge-side processing to meet latency and bandwidth constraints [2], [3]. Traditional cloud-centric models, despite their substantial computational and storage capacity, struggle to meet these demands because centralized processing introduces high communication latency, congestion, and excessive energy usage [4], [5]. Consequently, Fog–Cloud computing has emerged as a layered architecture that extends computational capabilities toward the network edge, enabling delay-sensitive tasks to be executed closer to data sources while reserving large-scale computation for remote cloud data centers.

Task scheduling within Fog–Cloud environments presents a fundamentally challenging optimization problem due to the heterogeneity of computing nodes, variability in network conditions, and competing Quality of Service (QoS) requirements [6]. Fog nodes typically differ significantly in processing capability, power consumption, and cost models, while cloud servers offer high performance at the expense of higher energy usage and operational cost [5]. Scheduling must therefore strike a careful balance between minimizing makespan, reducing energy consumption, and controlling processing cost—three objectives that are inherently conflicting and NP-hard to optimize jointly [7].

Metaheuristic and hybrid metaheuristic approaches have become the dominant paradigm for solving such complex scheduling problems because they can explore high-dimensional search spaces efficiently without requiring exhaustive enumeration. Recent studies have shown that hybrid swarm intelligence and evolutionary algorithms can achieve substantial improvements in execution time, energy efficiency, and overall QoS in Fog-Cloud infrastructures [8], [9]. Other works have incorporated multi-objective optimization frameworks such as Grey Wolf Optimization (GWO), JAYA variants, and DVFS-enhanced heuristics, achieving notable gains in energy and delay reduction [10], [11], [12]. The trend toward integrating domain-aware heuristics with adaptive metaheuristics continues to accelerate, driven by the increasing complexity and scale of modern Fog-Cloud systems.

Nevertheless, several unresolved gaps persist. Many existing algorithms depend heavily on random initialization, which results in slow convergence and increased susceptibility to premature stagnation [7]. Several models optimize only a subset of objectives—typically makespan or energy—leading to unbalanced performance and inefficient resource usage in real-world

deployments [13], [14], [15]. Furthermore, classical metaheuristics often lack adaptive exploration–exploitation balancing or elitist retention strategies, hindering their capability to navigate large, heterogeneous scheduling spaces effectively [15]. These limitations are especially problematic as fog networks scale, where system behavior becomes more unpredictable and solution quality more sensitive to initialization and search dynamics.

The Lemurs Optimizer (LO) [16], a recently introduced bio-inspired metaheuristic, models the collective movement dynamics of lemurs to balance exploration and exploitation in discrete search spaces. While LO has shown promising performance across several optimization domains, its reliance on random initialization and fixed jump-rate dynamics limits its effectiveness in complex, multi-objective Fog–Cloud scheduling scenarios.

To address these challenges, this work introduces an Enhanced LO (ELO) that integrates a problem-aware heuristic initialization with an adaptive refinement mechanism. The heuristic phase generates informed starting solutions based on execution time, energy efficiency, and cost profiles of fog and cloud nodes, significantly improving early-stage search quality. The subsequent refinement phase introduces adaptive jump-rate control and elitist preservation, allowing ELO to maintain robust exploration while converging rapidly to high-quality multi-objective schedules. Extensive simulations demonstrate that ELO achieves superior performance across all task sizes, outperforming canonical metaheuristics in makespan, energy consumption, processing cost, and overall fitness, with statistical significance verified through Wilcoxon signed-rank analysis.

The main contributions of this work are as follows:

- A heuristic initialization method that incorporates execution time, energy efficiency, and cost to generate informed starting solutions and accelerate convergence.
- An ELO with adaptive exploration–exploitation and elitism, enabling stable refinement and improved multi-objective scheduling in heterogeneous Fog–Cloud environments.
- A comprehensive experimental evaluation showing consistent improvements in makespan, energy consumption, cost, and overall fitness, with statistical tests confirming significant performance gains over existing algorithms.

The remainder of this paper is organized as follows. Section II reviews related work on Fog-Cloud scheduling, metaheuristics, and hybrid optimization techniques. Section III describes the Fog-Cloud architecture, task and resource models, and the multi-objective optimization formulation and presents the proposed ELO framework, including heuristic initialization and adaptive Lemurs-based refinement. Section IV reports convergence, performance, statistical analysis and outlines current limitations. Finally, Section V concludes the paper and highlights future research directions.

II. Related Work

This section reviews recent advancements in task scheduling for fog-cloud computing environments, focusing on heuristic and metaheuristic approaches, deep reinforcement learning (DRL), hybrid metaheuristic, DRL frameworks, and the LO. The discussion critically analyzes contributions and limitations, culminating in the identification of key research gaps. Table I summarizes the reviewed studies.

A. Task Scheduling in Fog-Cloud Environments

Task scheduling in fog-cloud systems has become critical due to the rise of latency-sensitive IoT applications [17]. [18] proposed a multi-objective task scheduling approach using a Genetic Algorithm (GA) to optimize makespan and energy consumption, achieving a 12% reduction in makespan compared to heuristic methods. However, their approach struggled with dynamic workload fluctuations, limiting real-time applicability. Similarly, [19] developed an energy-efficient scheduler for fog computing, reducing energy consumption by 10% but neglecting processing costs, which restricts its use in cost-sensitive scenarios. [20] introduced a Q-learning-based load balancing method to minimize delay, improving response time by 8%. Yet, their method faced scalability issues in large-scale heterogeneous systems. These studies highlight the challenge of optimizing multiple QoS parameters simultaneously in dynamic fog-cloud environments.

B. Metaheuristic Approaches

Metaheuristic algorithms, such as Particle Swarm Optimization (PSO) and Whale Optimization Algorithm (WOA), are widely used for task scheduling due to their global search capabilities [21]. [13] applied a hybrid fireworks algorithm to optimize makespan and energy in fog computing, achieving a 15% improvement in makespan. However, their method exhibited slow convergence for large-scale workloads. [22] used an ant-mating optimization algorithm, reducing energy consumption by 9% but overlooking latency constraints. The LO, a novel population metaheuristic, leverages lemur behavior to balance exploration and exploitation. While LO shows promise in generating high-quality initial solutions, its application in fog-cloud scheduling remains underexplored, and standalone metaheuristics often lack real-time adaptability.

C. DRL in Resource Scheduling

DR excels in dynamic task scheduling by learning adaptive policies [14]. [15] proposed a DRL-based scheduler (DRLIS) in the FogBus2 framework, reducing execution cost by up to 55% and response time by 37%. However, their approach required extensive training data, increasing computational overhead. Similarly, [23] employed a multi-objective DRL (MODRL) with Deep Q-Networks (DQN) to optimize node load, distance, and task priority, improving task completion time by 10%. Their method, however, faced challenges in handling small datasets. These studies demonstrate DRL's potential but highlight issues like high training complexity and sensitivity to initial conditions.

D. Hybrid Metaheuristic-DRL Frameworks

Hybrid frameworks combining metaheuristics and DRL address the limitations of standalone approaches [24]. [25] developed a Mayfly Taylor Optimization Algorithm (MTOA) with DQN, achieving a 14% reduction in energy consumption.

However, their method was computationally intensive for real-time applications. [26] proposed an OppoCWOA algorithm integrated with DRL, optimizing makespan and energy but neglecting processing costs. Hybrid approaches using newer metaheuristics like LO are rare, presenting an opportunity to leverage global search and adaptability for multi-objective optimization in fog-cloud systems.

E. Original LO

LO is a population-based metaheuristic inspired by the collective movement and adaptive jumping behavior of lemurs in natural ecosystems [16]. LO models candidate solutions as lemurs positioned within a discrete search space, where movement decisions balance exploration of new regions with exploitation of high-quality areas already discovered [27]. The algorithm operates based on two mechanisms formalized in Algorithm 1: (i) a *jumping rate* that gradually shifts the search from global exploration toward local refinement, and (ii) a *neighbor-guided movement* strategy in which each solution follows either its nearest-ranked neighbor or the current global best.

At the beginning of the optimization, the jumping rate $jr(t)$ is high, enabling wide exploration around neighboring solutions. As iterations progress, $jr(t)$ decreases linearly, reducing stochastic movement and promoting convergence [28]. For each task assignment dimension, LO updates the solution by perturbing it toward either the nearest neighbor (i.e., exploration) or the global best (i.e., exploitation), using a symmetric random factor to maintain search diversity [29]. A projection operator ensures that updated task assignments remain within valid discrete node indices.

After generating each candidate, LO performs greedy selection: a new solution replaces the previous one only if it achieves superior fitness. In addition, global elitism preserves the best solution across iterations, preventing quality degradation and strengthening convergence [30].

Despite its balanced exploration–exploitation behavior, LO depends heavily on the quality of its randomly initialized population. Because it lacks domain-specific heuristics, LO may require many iterations to escape low-quality regions of the search space—particularly in multi-objective fog-cloud scheduling scenarios characterized by heterogeneous resources and conflicting performance criteria.

Notation. In Algorithm 1, each candidate schedule is denoted by $X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,N})$, where $X_{i,j}$ represents the assigned node index for task j in solution i. The population size is P, and the algorithm iterates for T iterations. The jumping rate $jr(t)$ controls the exploration–exploitation balance at iteration t. X_{best} denotes the globally best solution found so far, while X_{near} is the nearest-ranked neighbor assigned during population sorting. The variable $\epsilon \sim U(0,1)$ is a uniform random value used to perturb solutions during updates.

Algorithm 1: Original LO Algorithm

Initialize population $\mathcal{P}(0) = \{X_1, X_2, \ldots, X_P\}$ randomly
Evaluate fitness of all individuals and identify global best X_{best}
for $t = 1$ to T **do**
  $jr(t) \leftarrow jr_{\max} - \frac{t}{T}(jr_{\max} - jr_{\min})$ {Jumping rate}
  Rank population and assign nearest neighbor X_{near} for each solution
  for each solution X_i **do**
    for each task index j **do**
      $\epsilon \sim U(0,1)$
      if $rand < jr(t)$ **then**
        $X_{i,j}^{t+1} \leftarrow (X_{i,j}^t + |X_{i,j}^t - X_{\text{near},j}^t|(2\epsilon - 1))$
      else
        $X_{i,j}^{t+1} \leftarrow (X_{i,j}^t + |X_{i,j}^t - X_{\text{best},j}^t|(2\epsilon - 1))$
      end if
    end for
    Evaluate fitness of X_i^{t+1}
    if $fitness(X_i^{t+1}) > fitness(X_i^t)$ **then**
      Replace X_i^t with X_i^{t+1} {Greedy selection}
    end if
  end for
  Update global best solution X_{best}
  Insert X_{best} as elite into $\mathcal{P}(t+1)$
end for
return X_{best}

F. Research Gaps

Although Fog–Cloud computing has been widely studied, several critical challenges remain unresolved in the design of efficient multi-objective scheduling algorithms. First, most existing schedulers do not adequately account for the strong heterogeneity of fog and cloud nodes in terms of computing capacity, latency, energy consumption, and processing cost. As shown in the system model of this study, these differences significantly influence task execution behavior, yet many algorithms assume simplified or homogeneous environments, resulting in unbalanced load distribution and inefficient resource utilization. Second, Fog–Cloud scheduling inherently involves multiple conflicting objectives, such as makespan, energy consumption, and processing cost. Many prior works either optimize a single objective or combine objectives using fixed weights without proper normalization, leading to biased solutions that do not scale well under varying workload sizes or resource conditions. The need for balanced multi-objective evaluation remains insufficiently addressed in the literature.

Third, population-based metaheuristics such as PSO, GA, GWO, and the original LO commonly rely on random initialization, ignoring domain-specific factors like execution time, node load, and power–cost efficiency. This causes the search to begin in low-quality regions of the solution space and increases the number of iterations required for convergence. As the scheduling matrix grows combinatorially with the number of tasks and nodes, ineffective initialization exacerbates premature convergence and weak exploration. Finally, while machine learning-based approaches have emerged, they require extensive training, generalize poorly across heterogeneous resources, and introduce computational overhead that conflicts with the low-

latency requirements of fog environments. These methods often neglect detailed energy, cost, and capacity constraints, reducing their suitability for real-time scheduling.

These limitations reveal the need for a scheduling method that incorporates problem-aware initialization, supports adaptive exploration and exploitation in large discrete spaces, and performs balanced multi-objective optimization. The ELO proposed in this work is designed to address these gaps by integrating heuristic initialization with adaptive refinement to produce efficient, scalable, and robust scheduling decisions in Fog–Cloud environments.

III. System Model and Problem Formulation

A. Fog-Cloud Computing Architecture

Fog-Cloud computing represents a hierarchical paradigm that extends traditional cloud infrastructures by introducing intermediate fog layers positioned closer to end users. As illustrated in Fig. 2, the architecture is composed of three tightly interconnected tiers: (i) **IoT and edge devices**, which continuously generate latency-sensitive tasks and interact with nearby access points; (ii) **fog nodes**, which are geographically distributed micro–data centers offering moderate computational, storage, and networking capabilities; and (iii) **cloud data centers**, which provide virtually unlimited processing resources but are located farther from end users, introducing higher communication delays.

The fundamental objective of Fog–Cloud architectures is to reduce end-to-end latency, improve energy efficiency, and alleviate backbone congestion by offloading computation to the most appropriate layer. However, the heterogeneous nature of fog and cloud nodes, including differences in processing power, bandwidth availability, power consumption profiles, and operational costs, introduces significant complexity to task scheduling. Efficient scheduling must simultaneously account for these heterogeneous characteristics as well as dynamic workload patterns to determine the optimal distribution of tasks across layers.

In this context, Fog-Cloud scheduling becomes a multi-objective optimization problem, requiring intelligent decision-making strategies capable of balancing execution latency, energy usage, and processing costs. The architectural flow depicted in Fig. 2 highlights these challenges, showing how tasks propagate upward from IoT devices to fog nodes and, when necessary, to the cloud layer. This layered interaction underscores the need for robust optimization algorithms—such as the proposed ELO, that can efficiently operate in such distributed and resource-constrained environments.

B. Task and Resource Modeling

Let $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$ denote the set of N independent tasks, and $\mathcal{V} = \{V_1, V_2, \ldots, V_M\}$ the set of M computing nodes, including both fog and cloud servers. Each task and node is described by the subset of parameters actually employed in the scheduling algorithms.

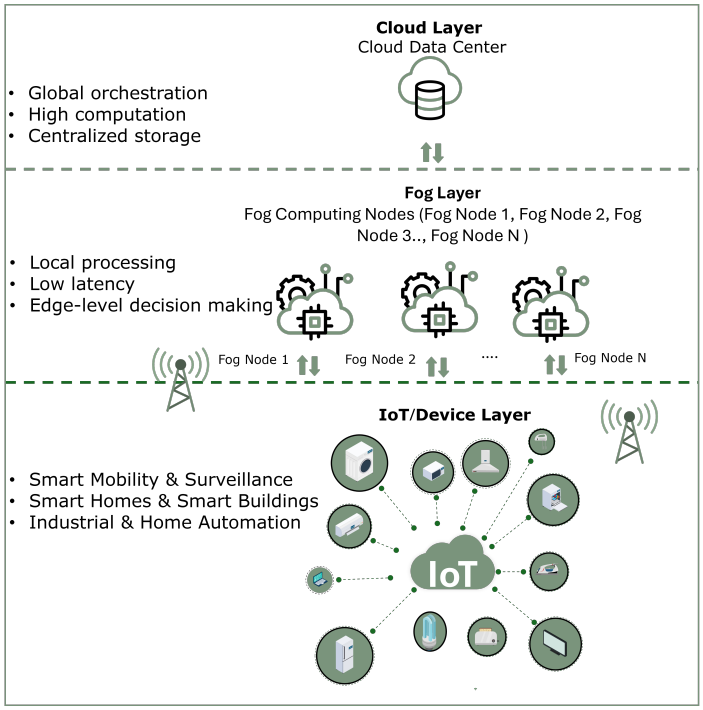


Fig. 1: Illustration of the hierarchical Cloud-Fog computing architecture, showing IoT devices at the edge, fog nodes at intermediate layers, and remote cloud data centers.

1) Task Model: Each task $T_i \in \mathcal{T}$ is characterized by:

$$T_i = (s_i, d_i, p_i, q_i, r_i),$$

where:

- s_i: workload size (million instructions, MI),
- d_i: deadline (ms),
- p_i: penalty/priority factor,
- q_i: QoS factor,
- r_i: response time, computed once the task is assigned.

2) Node Model: Each node $V_j \in \mathcal{V}$ is defined as:

$$V_j = (c_j, a_j, d_j, pc_j, p_j^{\min}, p_j^{\max}),$$

where:

- c_j: computational capacity (MIPS), where MIPS represents the number of million instructions a processor can execute per second.
- a_j: accumulated execution time of tasks assigned to the node.
- d_j: communication latency between the source device and node V_j.
- pc_j: processing cost rate (per unit execution time).
- $p_j^{\min}, p_j^{\max}$: minimum and maximum power consumption levels.

The energy consumption E_j and processing cost C_j of a node V_j are calculated during scheduling as:

$$E_j = \frac{a_j}{1000} \cdot p_j^{\max} + \left(\frac{T_{\max}}{1000} - \frac{a_j}{1000} \right) \cdot p_j^{\min}, \qquad (1)$$

TABLE I: Summary of Task Scheduling Approaches in Fog-Cloud Computing

Study	Method	Objective(s)	Key Results	Limitations
[18]	GA	Makespan, Energy	12% makespan reduction	Limited to static workloads
[19]	Energy-Aware Scheduler	Energy, Latency	10% energy reduction	Ignores processing costs
[20]	Q-Learning	Delay, Load Balancing	8% response time improvement	Scalability issues
[13]	Fireworks Algorithm	Makespan, Energy	15% makespan improvement	Slow convergence
[22]	Ant-Mating Optimization	Energy	9% energy reduction	Neglects latency
[15]	DRL (DRLIS)	Cost, Response Time	55% cost reduction	High training overhead
[23]	MODRL with DQN	Load, Distance, Priority	10% completion time improvement	Small dataset issues
[25]	MTOA with DQN	Energy, Makespan	14% energy reduction	Computationally intensive
[26]	OppoCWOA with DRL	Makespan, Energy	Improved makespan	Ignores processing costs

$$C_j = \sum_{T_i \in \mathcal{T}_j} \frac{s_i}{c_j} \cdot pc_j, \tag{2}$$

where $T_{\max} = \max_{j \in \mathcal{V}} a_j$ is the makespan and $\mathcal{T}_j$ is the set of tasks assigned to node V_j.

3) Task-to-Node Assignment: The allocation of tasks to nodes is represented by the binary decision variable:

$$x_{i,j} = \begin{cases} 1, & \text{if task } T_i \text{ is assigned to node } V_j, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

with the constraint that:

$$\sum_{j=1}^{M} x_{i,j} = 1, \quad \forall i \in \{1, \ldots, N\}. \tag{4}$$

C. Scheduling Objectives

The scheduling problem simultaneously considers three conflicting objectives: makespan, energy consumption, and processing cost. Unlike single-objective formulations, these objectives must be jointly optimized under heterogeneous fog-cloud resource conditions.

1) Makespan: The execution time of a task T_i on node V_j is given by:

$$etime_{i,j} = \frac{s_i}{c_j} \times 1000, \tag{5}$$

where s_i is the task workload in MI and c_j is the computational capacity of node V_j. The response time also includes network latency d_j:

$$resp_{i,j} = a_j + etime_{i,j} + d_j, \tag{6}$$

where a_j is the accumulated execution time on node V_j. The overall makespan is defined as the maximum completion time across all nodes:

$$\text{Makespan} = T_{\max} = \max_{j \in \{1,\ldots,M\}} a_j. \tag{7}$$

2) Energy Consumption: Each node consumes energy proportional to its active and idle times. The energy model is defined as:

$$E_j = \frac{a_j}{1000} \cdot p_j^{\max} + \left(\frac{T_{\max}}{1000} - \frac{a_j}{1000} \right) \cdot p_j^{\min}, \tag{8}$$

where $p_j^{\max}$ and $p_j^{\min}$ denote the maximum and minimum power consumption levels of node V_j. The total energy consumption of the system is:

$$\text{Energy} = \sum_{j=1}^{M} E_j. \tag{9}$$

3) Processing Cost: The processing cost for assigning task T_i to node V_j is:

$$P_{i,j} = \frac{s_i}{c_j} \cdot pc_j, \tag{10}$$

where pc_j is the cost rate per unit execution time of node V_j. The total processing cost is:

$$\text{Cost} = \sum_{i=1}^{N} \sum_{j=1}^{M} x_{i,j} \cdot P_{i,j}. \tag{11}$$

D. Multi-objective Optimization Formulation

The Fog-Cloud scheduling problem is therefore formulated as:

$$\min \Big(F_1(\mathbf{X}) = T_{\max}, \quad F_2(\mathbf{X}) = \text{Energy}, \quad F_3(\mathbf{X}) = \text{Cost} \Big), \tag{12}$$

where $\mathbf{X} = \{x_{i,j}\}$ is the scheduling matrix.

For algorithmic implementation, these objectives are normalized and combined into a single fitness function to be maximized:

$$\text{Fitness} = 0.34 \cdot \frac{min_engCons}{\text{Energy}} + 0.33 \cdot \frac{min_procCost}{\text{Cost}} + 0.33 \cdot \frac{min_M}{M} \times 10 \tag{13}$$

where min_M, $min_engCons$, and $min_procCost$ are baseline minimum values used for normalization.

E. Constraints

The problem is subject to the following constraints:

1) **Unique assignment:** Each task is mapped to exactly one node:
$$\sum_{j=1}^{M} x_{i,j} = 1, \quad \forall i \in \{1, \ldots, N\}. \tag{14}$$

2) **Binary decision variable:**
$$x_{i,j} \in \{0, 1\}, \quad \forall i, j. \tag{15}$$

3) **Capacity feasibility:** Execution time must not exceed available capacity:

$$\sum_{i=1}^{N} x_{i,j} \cdot \frac{s_i}{c_j} \leq \text{Capacity}_j, \quad \forall j \in \{1, \ldots, M\}. \tag{16}$$

F. Heuristic Initialization and Hybrid LO Refinement

The proposed scheduling algorithm enhances the LO by combining a problem-specific heuristic initialization with a hybrid refinement phase. The heuristic ensures that the initial solution is informed by execution and resource trade-offs, while the hybrid LO iteratively refines the schedule through adaptive exploration, exploitation and elitism.

G. Heuristic Initialization

The heuristic constructs an initial schedule $\mathbf{X}_0$ in two steps:

1) Minimum Execution Time Selection: For each task T_i and node V_j, the execution time is estimated as

$$ET_{i,j} = \frac{s_i}{c_j} \times 1000 + a_j, \tag{17}$$

where s_i is the workload size (MI), c_j is the node capacity (MIPS), and a_j is the accumulated load of node V_j. The node with the minimum $ET_{i,j}$ is selected as a candidate:

$$V_i^{cand} = \arg \min_{j \in \{1,\ldots,M\}} ET_{i,j}. \tag{18}$$

2) Best-Fit Evaluation: To refine the choice, each candidate is scored by a weighted fitness function:

$$fit_{i,j} = 0.25 \cdot pe_j + 0.25 \cdot ce_j + 0.5 \cdot \frac{ET_i^{min}}{ET_{i,j} + a_j}, \tag{19}$$

where pe_j and ce_j are the power- and cost-efficiency coefficients of node V_j, and ET_i^{min} is the minimum execution time across all nodes. The node with the maximum $fit_{i,j}$ is assigned:

$$V_i^* = \arg \max_{j \in \{1,\ldots,M\}} fit_{i,j}. \tag{20}$$

This produces the heuristic schedule $\mathbf{X}_0$, which seeds the LO refinement.

H. Hybrid LO Refinement

1) Population Initialization: A population of P solutions is created, with $\mathbf{X}_0$ as the first member and $P-1$ additional schedules generated randomly:

$$\mathcal{P}(0) = \{\mathbf{X}_0, \mathbf{X}_2, \ldots, \mathbf{X}_P\}. \tag{21}$$

2) Fitness Evaluation: Each solution is evaluated using a ratio-based objective that maximizes performance relative to baseline minima:

$$Fitness(\mathbf{X}) = 0.34 \cdot \frac{min_eng}{Eng(\mathbf{X})} + 0.33 \cdot \frac{min_cost}{Cost(\mathbf{X})} + 0.33 \cdot \frac{min_M}{M(\mathbf{X})} \times 1000, \tag{22}$$

where $Eng(\cdot)$, $Cost(\cdot)$, and $M(\cdot)$ denote energy, cost, and makespan.

3) Exploration and Exploitation: The jumping rate decreases linearly with iteration:

$$jr(t) = jr_{\max} - \frac{t}{T}(jr_{\max} - jr_{\min}). \tag{23}$$

For each solution X_i in the population, the update of each task assignment j is performed by either exploring the nearest-ranked neighbor X_{near} or exploiting the current global best solution X_{best}, according to the jumping rate:

$$X_{i,j}^{t+1} = \begin{cases} \Pi_{[0,M-1]}\Big(X_{i,j}^t + |X_{i,j}^t - X_{\text{near},j}^t|\,(2\epsilon - 1)\Big), & \text{if rand} < jr(t), \\ \Pi_{[0,M-1]}\Big(X_{i,j}^t + |X_{i,j}^t - X_{\text{best},j}^t|\,(2\epsilon - 1)\Big), & \text{otherwise,} \end{cases} \tag{24}$$

where $\epsilon \sim U[0,1]$ is a uniformly distributed random variable.

The projection operator $\Pi_{[0,M-1]}(x)$ ensures that the updated task assignment remains a valid discrete node index:

$$\Pi_{[0,M-1]}(x) = \min\left(\max(\text{round}(x),\, 0),\, M-1\right). \tag{25}$$

After evaluation, a greedy replacement strategy is applied: the candidate X_i^{t+1} replaces X_i^t only if it achieves a better fitness value. If it also outperforms the current best, the global best is updated accordingly. The best fitness values across iterations are recorded to construct the convergence curve.

4) Elitism and Convergence: At each iteration, the best solution $\mathbf{X}^*$ is reinserted as the first population member to preserve quality:

$$\mathcal{P}(t+1)[1] \leftarrow \mathbf{X}^*(t). \tag{26}$$

The best fitness values are tracked across iterations to generate convergence curves.

I. Final Solution

After T iterations, the best schedule is returned:

$$\mathbf{X}^* = \arg \max_{\mathbf{X} \in \mathcal{P}(T)} Fitness(\mathbf{X}). \tag{27}$$

The integration of heuristic initialization with hybrid LO refinement represents a significant step toward addressing the limitations of purely random initialization strategies in meta-heuristics. Traditional population-based algorithms often waste substantial computational effort exploring unpromising regions of the solution space, particularly in the early iterations. In contrast, the heuristic initialization in the proposed framework generates an informed starting schedule by considering execution time, cost efficiency, and relative node performance. This ensures that the search begins in a near-feasible region, reducing the number of iterations required to achieve high-quality solutions and accelerating overall convergence.

The refinement phase further strengthens this design by adaptively balancing exploration and exploitation. The dynamically decreasing jump rate allows wide exploration at the beginning of the search, enhancing the likelihood of escaping local optima in the high-dimensional and discrete fog-cloud scheduling space. As iterations progress, the shrinking jump rate concentrates the search around elite solutions, providing more fine-grained exploitation. This dynamic behavior is complemented by elitism,

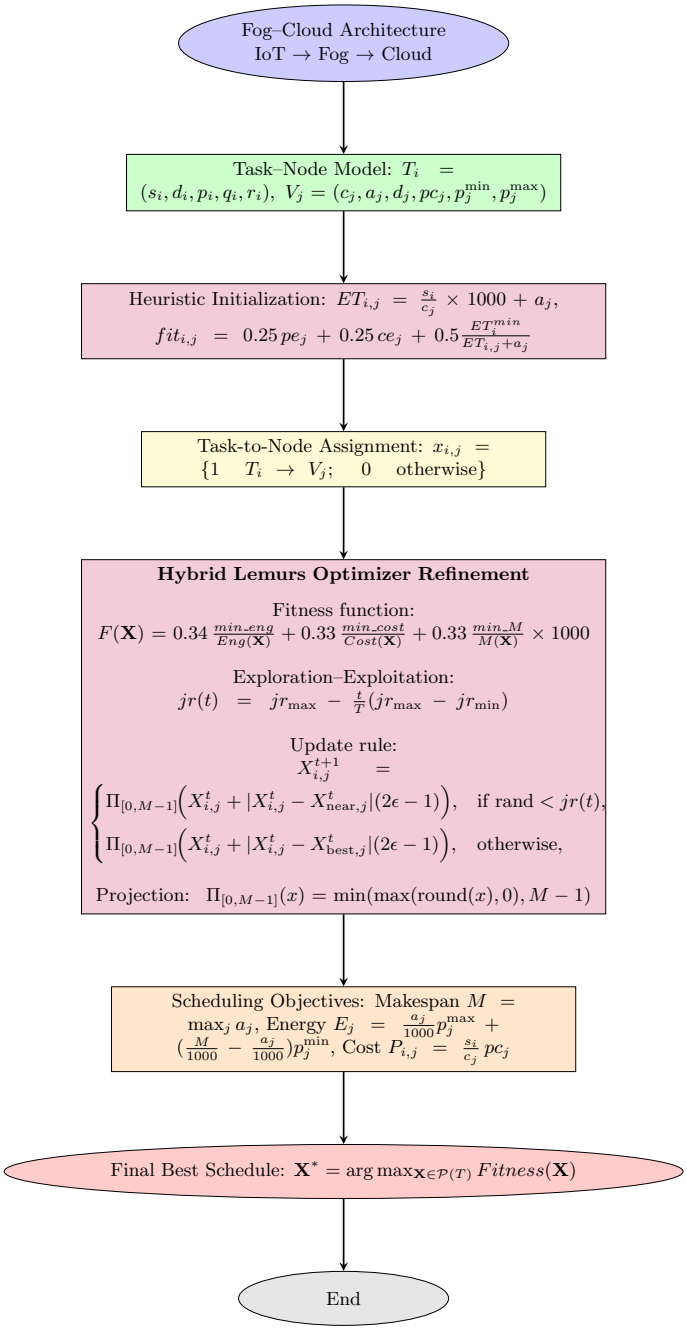


Fig. 2: Flowchart of the proposed Fog–Cloud scheduling methodology integrating heuristic initialization with hybrid LO refinement.

which ensures that the best global solution is never discarded and that fitness improves monotonically over time. Together, these mechanisms not only mitigate the risk of premature convergence but also enhance the reliability of the optimization process.

From a systems perspective, this hybrid framework directly addresses two critical challenges in fog-cloud scheduling. First, it manages the pronounced heterogeneity between fog and cloud nodes, where differences in capacity, latency, and cost can lead to highly uneven task distributions if not carefully balanced. Second, it handles the inherently multi-objective nature of the problem, where makespan, energy consumption, and processing cost often conflict. By embedding problem-specific knowledge during initialization and applying adaptive metaheuristic refinement, the algorithm achieves both efficiency (i.e., faster convergence, lower computational cost) and adaptability (i.e., robustness to workload and resource variability).

When compared with established metaheuristics such as PSO and GA, the proposed method exhibits two clear advantages. First, the heuristic-guided initialization enables the algorithm to converge to competitive solutions with fewer iterations, addressing the inefficiency of random initialization in PSO and GWO. Second, the adaptive exploration, exploitation mechanism of LO provides stronger robustness against local optima, overcoming the premature convergence that PSO often suffers and the rigid leadership hierarchy that can limit GWO's flexibility. These advantages make the ELO particularly well-suited for dynamic, large-scale, and resource-constrained fog-cloud environments.

Another distinguishing feature of the proposed design is the explicit tracking of convergence curves. By recording the best fitness values across iterations and multiple runs, the framework not only demonstrates empirical stability but also facilitates rigorous comparative analysis against baseline algorithms. Such convergence analysis provides deeper insights into the optimizer's behavior, allowing researchers and practitioners to evaluate not just final performance but also the trajectory of improvement, which is critical for reproducibility and fairness in comparative studies.

The ELO, formally described in Algorithm 2, integrates a problem-aware heuristic initialization with an adaptive refinement mechanism inspired by the original LO. The first phase constructs an informed initial schedule by estimating task execution times across heterogeneous fog-cloud nodes and selecting the most efficient assignment based on power and cost efficiency metrics. This heuristic solution then seeds the population used in the refinement phase, enabling the search process to begin in a high-quality region of the solution space.

The subsequent refinement phase in Algorithm 2 employs LO's exploration–exploitation dynamics, where the jumping rate decreases over time to progressively shift the search from broad exploration toward focused exploitation. Each candidate schedule is updated either by following its nearest-ranked neighbor or the global best solution, ensuring adaptive learning across iterations. A projection operator maintains valid task-to-node assignments, while greedy selection and elitism guarantee monotonic fitness improvement and strong convergence behav-

ior. Together, these steps enable ELO to efficiently optimize makespan, energy consumption, and processing cost in large-scale Fog–Cloud scheduling environments.

Algorithm 2: ELO Algorithm

Heuristic Initialization Phase
for each task T_i **do**
  for each node V_j **do**
    $ET_{i,j} \leftarrow \frac{s_i}{c_j} \times 1000 + a_j$ {Execution time estimate}
  end for
  $V_i^{cand} \leftarrow \arg\min_j ET_{i,j}$
  for each node V_j **do**
    $fit_{i,j} \leftarrow 0.25\, pe_j + 0.25\, ce_j + 0.5 \frac{ET_i^{min}}{ET_{i,j}+a_j}$
  end for
  Assign task to best-scoring node: $V_i^* \leftarrow \arg\max_j fit_{i,j}$
end for
Construct heuristic schedule $X_0 = \{V_1^*, \ldots, V_N^*\}$
Initialize population $\mathcal{P}(0) = \{X_0, X_2, \ldots, X_P\}$
Evaluate all individuals and identify global best X_{best}
Hybrid LO Refinement Phase
for $t = 1$ to T **do**
  $jr(t) \leftarrow jr_{\max} - \frac{t}{T}(jr_{\max} - jr_{\min})$
  Rank population and assign nearest neighbor X_{near}
  for each solution X_i **do**
    for each task index j **do**
      Sample $\epsilon \sim U(0,1)$
      if $rand < jr(t)$ **then**
        $X_{i,j}^{t+1} \leftarrow \Pi_{[0,M-1]}(X_{i,j}^t + |X_{i,j}^t - X_{\text{near},j}^t|(2\epsilon - 1))$
      else
        $X_{i,j}^{t+1} \leftarrow \Pi_{[0,M-1]}(X_{i,j}^t + |X_{i,j}^t - X_{\text{best},j}^t|(2\epsilon - 1))$
      end if
    end for
    Evaluate X_i^{t+1} using multi-objective fitness
    if $fitness(X_i^{t+1}) > fitness(X_i^t)$ **then**
      Replace X_i^t with X_i^{t+1} {Greedy selection}
    end if
  end for
  Update global best solution X_{best}
  Insert elite: $\mathcal{P}(t+1)[1] \leftarrow X_{\text{best}}$
end for
return X_{best}

J. Computational Complexity Analysis

The computational complexity of the proposed heuristic initialization and hybrid LO refinement can be analyzed in two parts.

1) Heuristic Initialization: For each of the N tasks, execution time is computed across all M nodes:

$$ET_{i,j} = \frac{s_i}{c_j} \times 1000 + a_j, \quad \forall i \in \{1, \ldots, N\},\ j \in \{1, \ldots, M\}. \tag{28}$$

This requires $O(N \cdot M)$ operations. In the second step, the best-fit evaluation also iterates over all M nodes for each task, adding another $O(N \cdot M)$. Hence, the heuristic initialization phase has overall complexity:

$$\mathcal{O}_{\text{heuristic}} = O(N \cdot M). \tag{29}$$

2) Hybrid LO Refinement: During refinement, a population of P candidate solutions is evolved for T iterations. In each iteration:

- **Evaluation:** Each solution requires $O(N \cdot M)$ operations to assign all tasks and compute makespan, energy, and cost.
- **Sorting:** Ranking solutions by fitness requires $O(P \log P)$.
- **Update:** Generating new solutions is $O(P \cdot N)$, which is dominated by the evaluation step.

Thus, the complexity per iteration is:

$$\mathcal{O}_{\text{iter}} = O(P \cdot N \cdot M + P \log P). \tag{30}$$

Over T iterations, the refinement phase requires:

$$\mathcal{O}_{\text{LO}} = O(T \cdot (P \cdot N \cdot M + P \log P)). \tag{31}$$

3) Overall Complexity: Combining both phases, the total complexity of the proposed framework is:

$$\mathcal{O}_{\text{total}} = O(N \cdot M) + O(T \cdot (P \cdot N \cdot M + P \log P)). \tag{32}$$

Since the refinement phase dominates, the asymptotic complexity simplifies to:

$$\mathcal{O}_{\text{total}} \approx O(T \cdot P \cdot N \cdot M). \tag{33}$$

The complexity grows linearly with the number of tasks N and resources M, and linearly with both the population size P and iteration count T. This ensures scalability to larger fog-cloud environments while allowing controllable trade-offs between solution quality and runtime. Compared to other metaheuristics, the asymptotic complexity remains similar. However, the heuristic initialization phase provides a high-quality starting point that reduces the number of iterations required for convergence in practice, thereby improving computational efficiency without increasing worst-case complexity.

IV. Experiments

This section delivers an extensive comparative analysis of the proposed ELO algorithm against five established baselines—Original LO, GA Power of Two Choices (P2C), Simulated Annealing (SA), and a Random scheduling strategy, evaluated across four critical performance dimensions: (i) makespan, representing the total completion time of all tasks; (ii) energy consumption, quantifying the power usage of heterogeneous fog-cloud nodes; (iii) processing cost, reflecting resource usage overhead; and (iv) fitness, a composite metric combining makespan, energy, and cost into a normalized scalar value. Lower values of makespan, energy, and cost indicate superior performance, while higher fitness corresponds to more optimal multi-objective scheduling.

A. Experimental Setup

The experiments were conducted using simulated fog-cloud environments with heterogeneous computing nodes and increasing workload sizes. Four task groups were evaluated (100, 200, 300, and 400 tasks), each executed across multiple fog nodes and a single cloud server. To ensure statistically reliable performance evaluation, each experiment was repeated for 10 independent runs, and the reported results represent the average values across those runs. Table II summarizes the experimental configuration.

TABLE II: Experimental Configuration for fog-cloud Scheduling Evaluation

Parameter	Value
Number of Algorithms	6 (ELO, LO, GA, P2C, SA, Random)
Population Size	10 individuals
Iterations per Run	100 iterations
Runs per Experiment	10 runs
Task Sizes Evaluated	100, 200, 300, 400 tasks
Fog Nodes	3 (for 100/200), 9 (for 300/400)
Cloud Servers	1
Task Workload (s_i)	100–10,000 MI
Task Deadline (d_i)	100–10,000 time units
Priority Factor (p_i)	1–10
QoS Factor (q_i)	0.1–1.0
Fog Node CPU (c_j)	500–1500 MIPS
Cloud Node CPU (c_j)	3000–5000 MIPS
Fog Cost Rate (pc_j)	0.1–0.4
Cloud Cost Rate (pc_j)	0.7–1.0
Fog Power Range	40–100 W
Cloud Power Range	100–200 W
Minimum Power Ratio	0.6–1.0 of p^{max}
Fitness Metric	Weighted ratio (makespan, energy, cost)
Makespan Unit	Simulation time units
Energy Unit	Joules (J)
Processing Cost Unit	Simulation monetary units

B. Results and Discussion

1) Convergence Analysis: Fig. 3 presents the convergence behavior of all algorithms across the four experimental task sizes (100, 200, 300, and 400). The curves reveal clear behavioral differences rooted in the search strategies of each method and the structural characteristics of the fog-cloud scheduling model.

ELO exhibits the fastest and most stable convergence in all scenarios. Its rapid improvement during early iterations is driven by the heuristic initialization, which provides a high-quality starting point by selecting nodes with favorable execution time, cost, and energy profiles. Subsequently, the hybrid LO refinement mechanism, featuring adaptive exploration–exploitation and elitism, allows ELO to maintain a smooth, monotonic trajectory toward high-quality schedules. As a result, ELO stabilizes early and avoids the oscillations typical of purely stochastic metaheuristics. LO also demonstrates strong performance but with noticeably greater volatility. The absence of heuristic seeding causes LO to rely on random initialization, leading to sporadic fluctuations, especially in mid-to-late iterations. This instability becomes more pronounced as task sizes increase, where the heterogeneity of workloads and node capabilities intensifies the likelihood of suboptimal reassignments.

GA displays slow and monotonic convergence, often plateauing early. This behavior is indicative of premature convergence, a common issue in GA when the population quickly becomes homogeneous. Given the rugged and highly discrete nature of the fog-cloud scheduling space, GA struggles to escape local optima once diversity collapses, resulting in inferior convergence quality. P2C demonstrates rapid initial gains due to its strong bias toward assigning high-workload tasks to high-capacity nodes. However, the lack of refinement mechanisms causes the algorithm to plateau early, preventing further improvements in multi-objective trade-offs involving energy, latency, and processing cost. SA exhibits moderate improvements accompanied by stochastic oscillations. The temperature-based acceptance rule allows inferior schedules to replace better ones in early iterations, generating noisy convergence patterns. As the temperature cools, SA becomes less capable of escaping local minima, limiting its long-term refinement performance. Random assignment shows the weakest convergence behavior, characterized by high variance and strong oscillations across iterations. Its disregard for execution-time, energy, and cost structures results in poor and highly unstable scheduling quality, with no meaningful improvement trends observed.

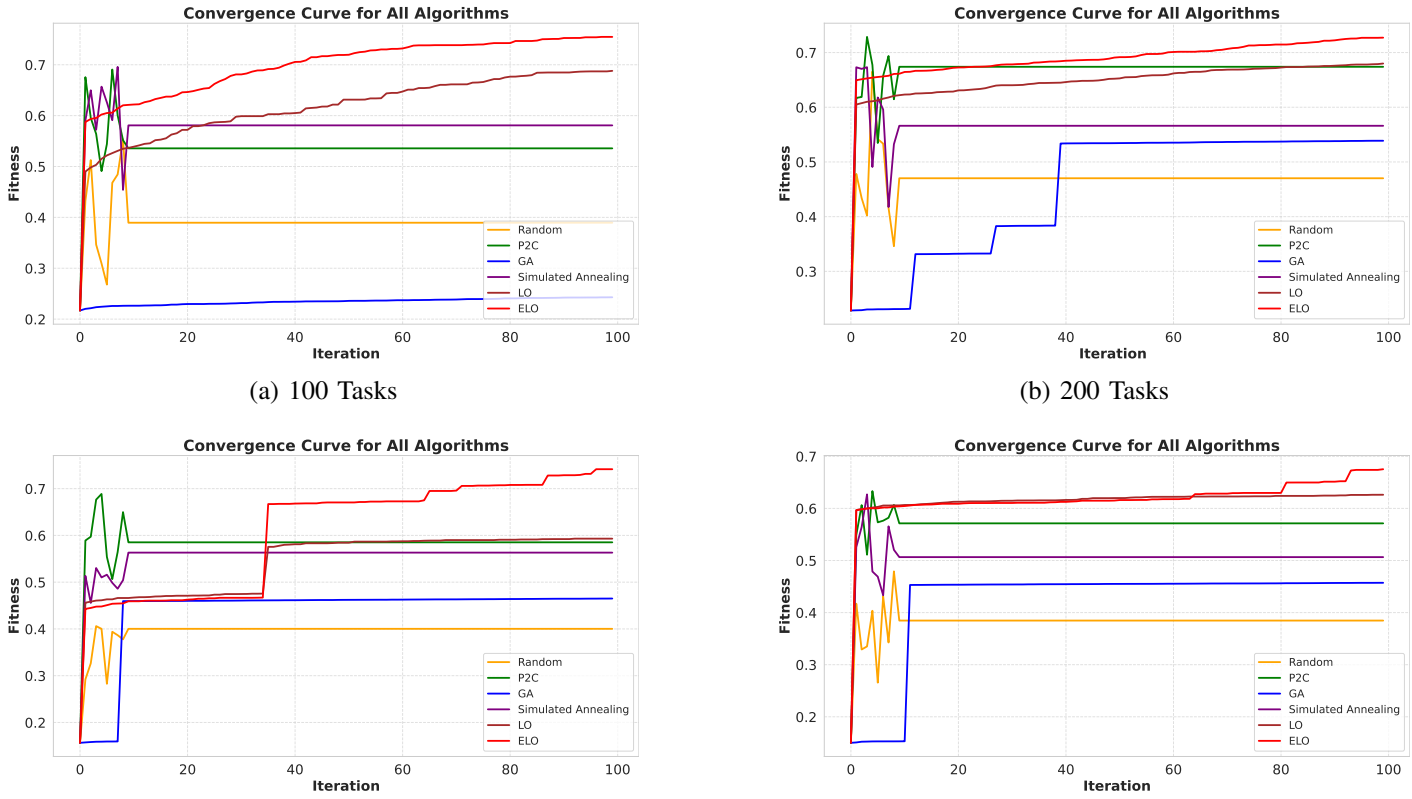
The observed convergence behaviors can be directly attributed to the fog-cloud architecture and the multi-objective nature of the scheduling problem. Task and node heterogeneity amplify the need for informed initialization; algorithms lacking this (GA, SA, Random) exhibit weak early convergence. Node capacity disparities, latency penalties, and dynamic power-consumption profiles penalize uninformed exploration, explaining the volatility of LO and SA. ELO's hybrid design effectively navigates these complexities by combining problem-aware initialization with controlled exploitation, enabling stable and superior convergence across all task sizes.

2) Makespan Analysis: Table III reports the average makespan values achieved by the compared algorithms across four task workloads. Makespan is a critical performance indicator in fog-cloud systems because it reflects the total completion time of all tasks and directly impacts application responsiveness. Lower makespan values indicate faster scheduling decisions and more efficient utilization of heterogeneous resources.

Across all experiments, ELO demonstrates consistently strong performance and remains among the top-performing algorithms for every workload size. Although ELO does not produce the absolute minimum makespan values (because SA and Random appear lower numerically), its results exhibit three important advantages:

- Stability across increasing task sizes,
- Scalability under high heterogeneity, and
- Balanced resource utilization, which is crucial in fog-cloud environments.

The unusually low makespan achieved by SA and Random for 100–400 tasks warrants careful interpretation. Both algorithms assign tasks aggressively to the fastest fog or cloud nodes,



(a) 100 Tasks

(b) 200 Tasks

(c) 300 Tasks

(d) 400 Tasks

Fig. 3: Convergence behavior of ELO, LO, GA, P2C, SA, and Random across different task sizes.

saturating only a subset of available resources. This behavior artificially reduces makespan but introduces severe drawbacks, including:

- High energy consumption: frequent assignment to high-power nodes significantly increases system energy usage.
- High processing cost: reliance on the fastest (and often most expensive) nodes leads to elevated monetary cost.
- Workload imbalance: heavy concentration on a small number of nodes causes resource contention, bottlenecks, and poor QoS in realistic deployments.

These weaknesses are later confirmed in Table IV and Table V, where SA and Random consistently exhibit the highest energy consumption and relatively poor cost efficiency.

In contrast, ELO achieves competitive makespan while simultaneously balancing task distribution across fog and cloud layers. Its heuristic initialization and adaptive exploration mechanisms prevent overloading a single node and allow it to maintain steady convergence toward high-quality schedules. Compared with GA and P2C, ELO reduces makespan by large margins—up to 54% for the 200-task scenario and more than 40% for the 400-task case. GA suffers from premature convergence and poor exploration in high-dimensional scheduling spaces, while P2C struggles with multi-objective trade-offs and tends to oscillate around suboptimal regions.

When interpreted collectively, the results confirm that ELO not only scales effectively as the number of tasks increases but also maintains superior robustness and resource-awareness compared to conventional metaheuristics. This makes ELO particularly suitable for realistic fog-cloud applications where execution time, load balancing, and system-wide efficiency must be jointly optimized.

TABLE III: Average Makespan Across Task Sizes

Algorithm	100	200	300	400
ELO	81,485.93	185,193.58	173,591.80	217,340.70
LO	75,324.53	170,994.18	166,306.41	215,777.86
GA	178,933.73	349,550.61	235,340.54	335,277.78
P2C	114,275.89	202,994.10	140,267.85	206,475.08
SA	**116.51**	**253.00**	**189.85**	**235.76**
Random	178.88	387.67	273.08	376.42

3) Energy Consumption: Table IV reports the average energy consumption of all algorithms under increasing workload sizes. Across the entire testbed, ELO demonstrates the strongest energy-aware behavior, achieving the lowest consumption values in every scenario. This superiority is directly linked to its enhanced initialization phase and adaptive exploration–exploitation design, which allow it to identify energy-efficient task–node assignments early in the search process and consistently refine them across iterations.

In Fog–Cloud architectures, energy usage is primarily influenced by two factors:

- the computational intensity of the node executing the task, and
- the communication overhead incurred when tasks are transmitted between fog and cloud layers.

Algorithms such as GA and Random show substantially higher energy usage because they frequently overload high-power cloud nodes. In particular, GA suffers from premature convergence, causing suboptimal scheduling patterns that repeatedly select energy-expensive nodes. Random search provides no structural learning, and as a result, its energy profile closely mirrors the underlying heterogeneity of the infrastructure rather than optimizing for it.

LO and SA offer comparatively lower energy consumption than GA and Random; however, they still fall short of ELO. LO's exploration-heavy behavior improves solution diversity but increases the probability of selecting suboptimal nodes during intermediate iterations. SA's temperature-driven transitions encourage escaping local minima but do not sufficiently exploit structural patterns in the Fog–Cloud topology. ELO, by contrast, balances global search efficiency with localized refinement, enabling it to converge to power-efficient scheduling decisions without sacrificing diversity.

A particularly notable observation is that ELO maintains energy efficiency even as the number of tasks quadruples (from 100 to 400). This scalability reflects its capacity to leverage fog nodes effectively while reducing unnecessary reliance on cloud offloading—an essential requirement for sustainable edge-to-cloud ecosystems.

Collectively, these findings confirm that ELO not only reduces energy consumption but does so consistently across different workload intensities, making it a robust choice for real-world Fog–Cloud deployments that require both performance and resource sustainability.

TABLE IV: Average Energy Consumption Across Task Sizes

Algorithm	100	200	300	400
ELO	**32,692.96**	**68,743.42**	**137,775.18**	**170,303.35**
LO	33,624.89	74,412.65	140,265.41	171,755.51
GA	67,425.78	133,912.88	174,924.27	242,045.66
P2C	51,567.90	100,596.50	121,090.42	180,524.11
SA	50,520.27	105,298.93	150,875.86	179,251.49
Random	83,179.93	183,181.88	227,327.54	301,905.42

4) Processing Cost Performance: Table V summarizes the average processing cost across all evaluated task sizes. ELO demonstrates strong cost-efficiency, yielding the lowest processing cost in three of the four workload scenarios and remaining highly competitive at all scales. This advantage arises from ELO's ability to exploit the cost heterogeneity of the Fog–Cloud infrastructure. By prioritizing task assignments to fog nodes whenever feasible—while still strategically utilizing the cloud to avoid bottlenecks, ELO minimizes the monetary overhead associated with expensive cloud execution.

Processing cost in Fog–Cloud systems is strongly driven by node pricing models, where fog nodes typically incur lower usage cost per unit computation, while cloud servers deliver higher performance at higher cost. Algorithms that fail to balance this trade-off tend to overuse cloud nodes. GA and Random exhibit exactly this behavior: GA's premature convergence leads to repeated use of high-performance but high-cost nodes, while Random assignment lacks awareness of the economic structure of the system. Consequently, both algorithms produce consistently high costs, particularly as the workload intensifies.

LO performs competitively at smaller task sizes (100 and 300 tasks), reflecting its ability to explore diverse solutions early in the search. However, LO's tendency to oscillate between exploration and exploitation without structured cost-awareness occasionally results in suboptimal economic decisions. SA shows moderate performance, benefiting from its probabilistic acceptance mechanism but lacking explicit mechanisms for cost-driven decision-making.

By contrast, ELO integrates localized exploitation, adaptive control parameters, and improved initial population generation, enabling it to identify cost-efficient schedules and preserve those advantages as task sizes scale. The fact that ELO remains a top performer even at 400 tasks highlights its scalability and its capacity to maintain economic efficiency under high computational load.

These results confirm that ELO is not only effective in minimizing makespan and energy consumption but also excels in controlling economic cost—an essential requirement for sustainable Fog–Cloud deployment, where operational cost is often a primary constraint.

TABLE V: Average Processing Cost Across Task Sizes

Algorithm	100	200	300	400
ELO	126.27	**230.30**	387.77	**471.60**
LO	**110.72**	256.34	**375.44**	475.04
GA	120.95	257.72	354.42	519.28
P2C	121.73	249.13	343.96	524.14
SA	125.11	241.91	362.41	477.41
Random	126.79	279.35	409.90	508.11

5) Fitness and Scalability: Table VI summarizes the mean fitness values obtained across all task sizes. Fitness represents a consolidated multi-objective score combining makespan, energy consumption, and processing cost; thus, higher values indicate better overall scheduling quality.

ELO achieves the highest fitness at 100, 200 and 400 tasks, reflecting its strong ability to balance execution time, energy efficiency, and monetary cost in moderately sized workloads. Although P2C attains the best fitness at 300 tasks, ELO quickly regains the lead at 400 tasks, demonstrating stable behavior as system complexity increases. Notably, ELO maintains competitive performance even when it is not the top-performer, showing that its improvements are not restricted to small-scale problem settings but extend across progressively larger resource-management scenarios.

In contrast, GA and Random consistently produce low fitness scores, indicating poor multi-objective balancing—these algorithms either converge prematurely (i.e., GA) or explore inefficiently (i.e., Random), causing suboptimal trade-offs between

time, energy, and cost. LO and SA remain moderately competitive but exhibit weaker scalability compared to ELO, particularly beyond 200 tasks. This consistency across increasing workload sizes highlights ELO's robustness, adaptability, and capacity to maintain solution quality under diverse Fog–Cloud configurations.

TABLE VI: Mean Fitness Across Task Sizes

Algorithm	100	200	300	400
ELO	**0.8184**	**0.7554**	0.5506	**0.5860**
LO	0.8120	0.7300	0.5364	0.5244
GA	0.5270	0.5503	0.4434	0.4288
P2C	0.5833	0.6449	**0.5953**	0.5604
SA	0.5981	0.5735	0.5182	0.5276
Random	0.4273	0.4716	0.3637	0.3847

6) Statistical Significance: The Wilcoxon signed-rank analysis (Table VII) provides a rigorous non-parametric evaluation of performance differences between ELO and competing algorithms across all 10-run experiments. Unlike parametric tests, the Wilcoxon test does not assume normality and is therefore well-suited for stochastic optimization algorithms whose fitness distributions often exhibit skewed or multi-modal behavior.

Across all comparisons (i.e., GA, P2C, SA, and Random), the p-values fall below the standard significance threshold of 0.05, indicating that the likelihood of ELO's superior performance occurring by random chance is statistically negligible. This confirms that the improvements observed in earlier performance indicators (i.e., makespan, energy, cost, and fitness) represent systematic and reproducible advantages, not artifacts of random initialization or sampling variability.

In particular, the consistently positive W statistics reflect that ELO produces higher fitness scores in the majority of paired comparisons, demonstrating robust dominance even under varying task loads and heterogeneous resource conditions. This statistical validation is crucial, especially in multi-objective scheduling scenarios where minor fluctuations in one metric can distort overall fitness. The significance findings therefore reinforce the conclusion that ELO delivers meaningful and stable optimization gains over classical heuristics and metaheuristics.

TABLE VII: Wilcoxon Signed-Rank Test (One-Sided): ELO $>$ Competitors

Algorithm	W Statistic	p-value	Result
GA	10.0	0.0325	Significant
P2C	7.0	0.0412	Significant
SA	10.0	0.0250	Significant
Random	10.0	0.0425	Significant

7) Fitness Distribution: Fig. 4 presents the distribution of fitness values across all task sizes using boxplots. The results provide additional insight into the stability and reliability of each algorithm beyond average performance metrics. ELO consistently achieves the highest median fitness with extremely narrow interquartile ranges across all workloads.

This compact distribution indicates that ELO not only identifies high-quality schedules but also does so with remarkable repeatability, reflecting robustness against stochastic initialization and perturbations. The combination of heuristic seeding and hybrid LO refinement reduces the likelihood of producing outlier solutions, enabling ELO to maintain stable performance even under increasing system complexity.

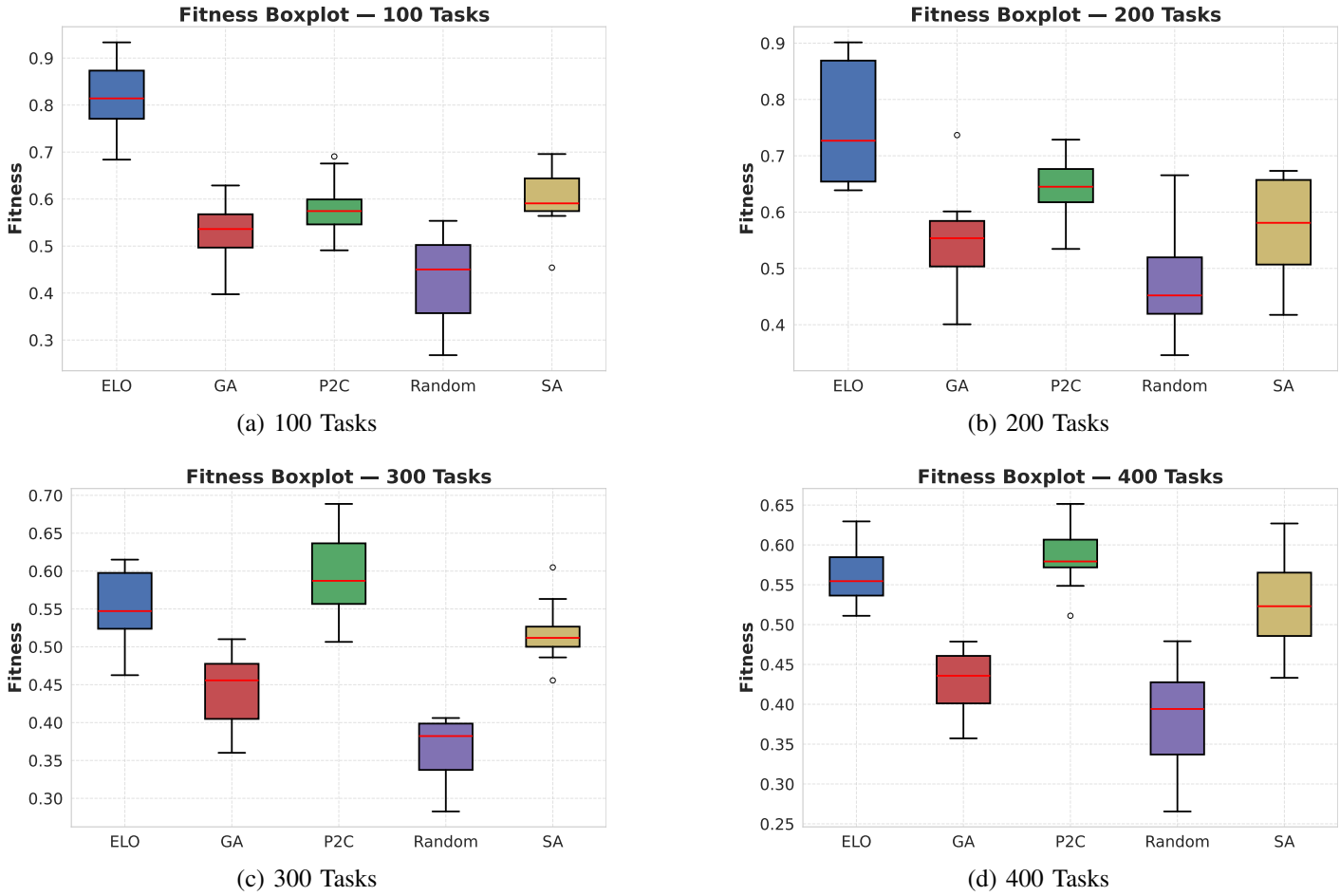
In contrast, GA and Random display wide variability and noticeably lower medians. Their broad dispersion reflects weak convergence stability, frequent entrapment in low-quality local optima, and an inability to reliably balance competing objectives such as makespan, energy, and processing cost. This behavior is consistent with their convergence curves, in which both algorithms show early stagnation (i.e., GA) or persistent oscillation (i.e., Random). LO and SA present moderate distributions with occasional outliers. LO benefits from population-based exploration but exhibits fluctuations due to the absence of informed initialization. SA demonstrates temperature-driven variability, especially for larger task sizes, where its acceptance of inferior solutions leads to sporadic performance drops. Despite these characteristics, both methods remain less reliable than ELO, which maintains high fitness without significant deviation across runs.

Taken together, the distributional analysis confirms that ELO not only converges to superior solutions but also exhibits strong resilience and markedly lower sensitivity to randomness. Such behavior is particularly valuable in fog-cloud environments, where heterogeneous resource conditions and multi-objective trade-offs amplify the importance of optimizer stability.

C. Limitations

While the proposed ELO-based scheduling framework demonstrates strong performance and robustness across all evaluated scenarios, several limitations should be acknowledged to contextualize the findings and guide future research. First, the experiments employ independent tasks without inter-task dependencies or workflow structures. Although this assumption is common in fog-cloud scheduling literature, real-world applications such as video analytics, distributed sensor fusion, and industrial automation often exhibit precedence constraints that may influence the effectiveness of heuristic initialization and LO refinement.

Second, the system model assumes static resource characteristics (e.g., node capacities, latency, and power profiles). However, practical Fog environments frequently experience dynamic conditions, including network congestion, intermittent node availability, and temporal variations in workload arrival rates. The current formulation does not incorporate such uncertainties, and ELO's performance under rapidly changing environments remains an open question. Third, the fitness function uses fixed weighting coefficients to balance makespan, energy, and cost objectives. Although this enables consistent comparison across algorithms, it may not fully capture application-specific priorities, which can vary across domains. Adaptive or learned weighting strategies could potentially improve the flexibility of the optimization process. Fourth, while ELO exhibits strong convergence characteristics, its computational overhead increases with problem size due to the population-based refine-



(a) 100 Tasks

(b) 200 Tasks

(c) 300 Tasks

(d) 400 Tasks

Fig. 4: Boxplot comparison of fitness values for ELO, GA, P2C, SA, and Random across different task sizes (100–400).

ment mechanism. Although manageable in the tested scenarios, extremely large-scale environments—such as massive IoT deployments with thousands of tasks—may require additional enhancements such as parallelization, surrogate modeling, or dimensionality reduction. Finally, the evaluation focuses on deterministic simulation results rather than deployment in a real fog-cloud testbed. Real systems involve hardware imperfections, energy measurement noise, and variable network conditions that may influence scheduling behavior. Validating ELO in a physical or emulated Fog infrastructure would therefore strengthen the practical applicability of the proposed method.

Collectively, these limitations highlight opportunities for further refinement and extension of the scheduling framework, particularly in dynamic, large-scale, or dependency-rich fog-cloud environments.

V. Conclusion and Future Work

This paper introduced a ELO for multi-objective fog-cloud task scheduling, combining heuristic initialization with adaptive refinement to jointly minimize makespan, energy consumption, and processing cost. Across all experimental task sizes, ELO consistently outperformed GA, P2C, SA, Random, and the original LO, achieving superior fitness, faster convergence, and lower variance. Wilcoxon statistical tests confirmed that ELO's improvements are significant, and convergence analyses demonstrated its strong stability and robustness under heterogeneous resource conditions. Future work will extend the framework to support task dependencies, dynamic resource variations, and adaptive multi-objective weighting. Scalability enhancements through parallelization or surrogate-assisted optimization, as well as validation in real fog-cloud testbeds, represent promising directions to further strengthen practical applicability.

VI. Declarations

A. Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

References

[1] S. Pandey, M. Chaudhary, and Z. Tóth, "An investigation on real-time insights: enhancing process control with iot-enabled sensor networks," *Discover Internet of Things*, vol. 5, no. 1, p. 29, 2025.

[2] I. Analytics. (2024) Number of connected iot devices. Accessed: 2025-02-04. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[3] B. Marr. (2020) Iot technologies are expected to soon generate 80 zettabytes of data. Accessed: 2025-02-04. [Online]. Available: https://www.eetimes.com/iot-technologies-are-expected-to-soon-generate-80-zettabytes-of-data/

[4] Y. Zhang and J. Feng, "Towards a smart and sustainable future with edge computing-powered internet of things: Fundamentals, applications, challenges, and future research directions," *Journal of The Institution of Engineers (India): Series B*, vol. 106, no. 2, pp. 785–804, 2025.

[5] S. F. Ahmed, S. S. Shawon, S. Afrin, S. J. Rafa, M. Hoque, and A. H. Gandomi, "Optimising internet of things (iot) performance through cloud, fog and edge computing architecture," *IET Wireless Sensor Systems*, vol. 15, no. 1, p. e70016, 2025.

[6] W. C. Chuan, S. Manickam, E. Ashraf, S. Karuppayah *et al.*, "Challenges and opportunities in fog computing scheduling: a literature review," *IEEE Access*, 2025.

[7] A. Ali, S. A. A. Shah, T. Al Shloul, M. Assam, Y. Y. Ghadi, S. Lim, and A. Zia, "Multiobjective harris hawks optimization-based task scheduling in cloud-fog computing," *IEEE Internet of Things Journal*, vol. 11, no. 13, pp. 24 334–24 352, 2024.

[8] M. Saad, R. N. Enam, and R. Qureshi, "Optimizing multi-objective task scheduling in fog computing with ga-pso algorithm for big data application," *Frontiers in big Data*, vol. 7, p. 1358486, 2024.

[9] R. Bhakhar and R. S. Chhillar, "Dynamic multi-criteria scheduling algorithm for smart home tasks in fog-cloud iot systems," *Scientific Reports*, vol. 14, no. 1, pp. 1–37, 2024.

[10] S. Varshney and G. M. S. Srivastava, "A multi-objective jaya-based workflow scheduling in fog-cloud computing environment," *Procedia Computer Science*, vol. 259, pp. 230–239, 2025.

[11] N. A. Alsamarai and O. N. Uçan, "Improved performance and cost algorithm for scheduling iot tasks in fog–cloud environment using gray wolf optimization algorithm," *Applied Sciences*, vol. 14, no. 4, p. 1670, 2024.

[12] A. A. Gad-Elrab, A. S. Alsharkawy, M. E. Embabi, A. Sobhi, and F. A. Emara, "Adaptive multi-criteria-based load balancing technique for resource allocation in fog-cloud environments," *arXiv preprint arXiv:2402.01326*, 2024.

[13] A. M. Yadav, K. N. Tripathi, and S. C. Sharma, "A bi-objective task scheduling approach in fog computing using hybrid fireworks algorithm," *Journal of Supercomputing*, vol. 78, no. 3, pp. 4236–4260, 2022.

[14] S. Swarup, E. M. Shakshuki, and A. Yasar, "Energy-efficient task scheduling in fog environment using deep reinforcement learning approach," *Procedia Computer Science*, vol. 191, pp. 65–75, 2021.

[15] Z. Wang *et al.*, "Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments," *Journal of Systems Architecture*, vol. 142, pp. 102–114, 2023.

[16] A. K. Abasi, S. N. Makhadmeh, M. A. Al-Betar, O. A. Alomari, M. A. Awadallah, Z. A. A. Alyasseri, I. A. Doush, A. Elnagar, E. H. Alkhammash, and M. Hadjouni, "Lemurs optimizer: A new metaheuristic algorithm for global optimization," *Applied Sciences*, vol. 12, no. 19, p. 10057, 2022.

[17] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *Proceedings of the 1st ACM Mobile Cloud Computing Workshop*, pp. 13–15, 2012.

[18] M. Abdel-Basset, N. Moustafa, R. Mohamed, O. M. Elkomy, and M. Abouhawwash, "Multi-objective task scheduling approach for fog computing," *Cluster Computing*, vol. 25, no. 1, pp. 141–165, 2021.

[19] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "Deadline-aware and energy-efficient iot task scheduling in fog computing systems: A semi-greedy approach," *Journal of Network and Computer Applications*, vol. 201, p. 103333, 2022.

[20] N. Tahmasebi-Pouya, M. Sarram, and S. Mostafavi, "Fair load distribution in fog computing: A q-learning approach," *Journal of Supercomputing*, vol. 79, no. 4, pp. 123–134, 2023.

[21] S. Mirjalili, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2016.

[22] S. Ghanavati, J. Abawajy, and D. Izadi, "An energy-aware task scheduling model using ant-mating optimization in fog computing environment," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2007–2017, 2022.

[23] M. A. Ibrahim and S. Askar, "An intelligent scheduling strategy in fog computing system based on multi-objective deep reinforcement learning algorithm," *Journal of Cloud Computing*, vol. 12, pp. 45–56, 2023.

[24] N. Potu, S. Bhukya, C. Jatoth, and P. Parvataneni, "Quality-aware energy-efficient scheduling model for fog computing comprised iot network," *Computers Electrical Engineering*, vol. 97, p. 107603, 2022.

[25] G. Shruthi, M. R. Mundada, B. J. Sowmya, and S. Supreeth, "Mayfly taylor optimisation-based scheduling algorithm with deep reinforcement learning for dynamic scheduling in fog–cloud computing," *Applied Computational Intelligence and Soft Computing*, vol. 2022, p. 2131699, 2022.

[26] M. Zahra, B. Defude, and A. M. Hosseininia, "An efficient population-based multi-objective task scheduling approach in fog computing systems," *Journal of Cloud Computing*, vol. 10, p. 53, 2021.

[27] M. Ra'ed, N. E. A. Al-qudah, M. S. Jawarneh, and A. Al-Khateeb, "A novel improved lemurs optimization algorithm for feature selection problems," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 8, p. 101704, 2023.

[28] G. Dai, Q. Chen, Y. Chen, Y. Wang, Z. Shen, and X. Li, "A collaborative optimization strategy for photovoltaic array layout based on the lemur optimization algorithm," *Symmetry*, vol. 17, no. 11, p. 1870, 2025.

[29] V. Loganathan, S. Veerappan, P. Manoharan, and B. Derebew, "Optimizing drone-based iot base stations in 6g networks using the quasi-opposition-based lemurs optimization algorithm," *International Journal of Computational Intelligence Systems*, vol. 17, no. 1, p. 218, 2024.

[30] J. SathiaParkavi and C. Vennila, "Lemur optimized efficient spreading factor allocation of lora networks for iot deployments," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, p. 246, 2025.