

Original Research

Received 05 July 2026

Revised 30 July 2026

Accepted 12 Aug 2026

Natural Resources for Human Health 2026, Vol. 6 (18s): 12–30

KEYWORDS:

Cloud Computing, Energy Efficiency, Fog Computing, Multi-Objective Optimization, Internet of Things (IoT), Smart Waste Management

Adaptive Multi-Objective Energy-Efficient Resource Allocation Strategy for Cloud-Fog Environment for Smart Waste Monitoring System

Pooja Rani Agarwal¹, Komal Alwani², Amit Kumar Chaturvedi³

¹Department of Computer Science, Bhagwant University, Ajmer, India

²Department of Computer Science, Bhagwant University, Ajmer, India

³Department of Computer Application, Government Engineering College, Ajmer, India

Corresponding Author Email: 1pooja.itindia33@gmail.com

Co-Author Email: 2komalrocks2@gmail.com, 3amit0581@gmail.com

ABSTRACT: Internet of Things (IoT) based smart waste bin systems in urban environments have revived the interest in processing the waste data with low latency and cost driven by the concept of Cloud-Fog computing architectures. Existing strategies on road fog resource allocation are mainly aimed at minimising response time (and cost), while not prioritising energy as a key concern, limiting the viability of the strategies at city level. This paper presents an Adaptive Multi-Objective Energy-Efficient Resource Allocation (AMOEERA) strategy for Cloud-Fog based smart waste monitoring, which optimizes response time (time delay), energy and cost together, based on an adaptive scheduling model of urgency-aware scheduling, under a five-layer IoT-Fog-Cloud architecture (IoT/Edge, Fog, Proxy/Gateway, Cloud, and Management/Application). The performance of the AMOEERA is compared with the First-Come-First-Served (FCFS), Particle Swarm Optimization (PSO) and Non-dominated Sorting Genetic Algorithm II (NSGA-II) on the following parameters: Response Time, Energy, Cost, Resource Utilization and Throughput. The strategies were tested in five device-scaling configurations (50-500 end devices) for 12 seeds of each strategy under a discrete-event simulation parameterized following an iFogSim-based fog/cloud model; the differences between the strategies were evaluated via the paired Wilcoxon signed-rank test. Results show that the AMOEERA algorithm can generate an average of 9.9% energy saving and 26.0% increment of resource utilization and 12.1% increment of throughput (which are statistically significant $p < 0.01$) with respect to NSGA-II algorithm in most of the configurations. The response time is comparable to NSGA-II with a marginal improvement in the former at smaller scales and the cost premium is small and largest at smaller scales but closing rapidly towards equality at 500 devices, the scheduling overhead is 241-370 times slower, decreasing at larger scales. The results also show the benefit of the lightweight, urgency-aware, greedy scheduler over a population-based optimizer with respect to response time and cost at scale, as well as energy efficiency/energy usage, utilization and throughput at a relatively low level of overhead; beneficial for, say, a city-wide deployment of smart waste monitoring, which would likely be large but not massive (as in 500 devices were tested here, and the observed trends suggest that those advantages would remain at larger scale/deployments).

1. INTRODUCTION

Rapid urbanization, population growth and industrial development are causing the amount of municipal solid waste (MSW) produced to grow. The amount of municipal solid waste generated globally is estimated to be around 3.4 billion tonnes per year by 2050, up from 2.01 billion tonnes in 2016 [1]. This unprecedented expansion puts increased strain on the municipal governments to enhance waste collection and management systems' efficiency, sustainability and cost-effectiveness. Typical waste collection is by a scheduled timetable in which waste collection vehicles are automatically guided to waste bins irrespective of how they are filled. This type of static collection methods can lead to unnecessary fuel use, more greenhouse gas emissions, traffic congestion, inefficient use of collection vehicles, and in some cases, waste bins overflowing creating a negative impact on the health of the local population and on the urban environment [2], [3]. With the advent of the internet of things (IoT), it is now possible to create smart systems for waste monitoring that can be used to get live information on waste bins. Modern smart bins with ultrasonic, gas, temperature sensors, and wireless communication modules are continuously measuring the fill level and environmental conditions and communicating the condition to the centralized management platforms [4], [5]. The real-time sensing function allows for a shift from a scheduled waste collection to a demand-driven waste management approach, which enhances the efficient management of the city's waste operations. A few studies have shown that implementing real-time sensing and intelligent vehicle routing can help to enhance the operational efficiency and minimize unnecessary collection trips [6]. However, route optimization only solves part of the problem, that of transport, and cannot be used to solve the computational problem of efficiently handling the vast amount of data generated by the IoT. Asian governments and societies are actively promoting digital, IoT-enabled waste management: China's 2019 zero-waste city pilot (11 cities, 5 regions) and 2025 waste-classification mandate, with IoT traceability already in use [6], [7]; Japan's 2024 plan promoting digital technologies and public awareness (90% target) [8]; and South Korea's volume-based fee (1995) with RFID-tracked food-waste billing [8]. In India, SBM-Urban 2.0 targets 100% door-to-door collection and scientific processing with real-time digital monitoring [9], and Viksit Bharat 2047 will require reliable waste databases and digital infrastructure as urban waste approaches about 435 million tonnes by 2050 [10].

In current smart waste monitoring systems, the sensor data are directly sent to cloud servers for processing and storage. While cloud computing provides an abundance of computing power and scalable storage, centralized processing creates communication latency, a greater bandwidth usage, and potential congestion, especially if there are many geographically dispersed IoT devices providing continuous data streams. For events like overflowing bins, hazardous gas leaks or fire risk detection, where a response must be made in a timely fashion, these limitations come into play. Fog computing has emerged as a viable solution to address these challenges as it enables the deployment of an intermediate computing layer between the IoT devices and the cloud data centers. By placing fog nodes near data sources, local processing, filtering and decision making can be carried out locally, which not only reduces the delay of communication but also reduces the amount of communication traffic, while reserving the computing resources on the cloud for high-computation, large-storage and city-wide optimization. While the adoption of fog computing greatly enhances system responsiveness, it also poses a new research challenge: resource allocation. Each task received from the IoT application needs to be checked if it should be executed in the fog node where it has been received and migrated to other fog nodes or sent to the cloud to be executed. The scheduling decision has a direct impact on various conflicting performance goals, including response time, energy usage, execution cost, network utilization, and QoS. Simply reducing the response time may result in too many nearby fog nodes being overloaded and reducing the energy consumption may lengthen execution time by clustering tasks on fewer resources. Likewise, reducing the operational cost could lead to over-dependence on cloud resources, which can lead to high communication delays. In this context, the problem of resource allocation in IoT-Fog-Cloud environments is a challenging multi-objective optimization problem [11].

In recent years, a number of resource allocation and task scheduling strategies are proposed using heuristic, metaheuristic, and evolutionary optimization techniques, such as First-Come, First-Served (FCFS), Round Robin, Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Non-dominated Sorting Genetic Algorithm II (NSGA-II) [12], [13]. These methods have been shown to achieve better results in one or two particular performance metrics or aggregate multiple metrics into a single fitness function with weights. The weighting of these factors must be done in a handcrafted manner, and can mask the conflicts of adding priorities. Moreover, a few studies address the energy-efficient resource allocation problem specifically for smart waste monitoring systems with IoT-enabled devices in the cloud-fog environment. The framework (QAIoET) recently proposed in [14] is one of the important approaches to optimize the resource allocation in smart waste monitoring at the QoS level, but it is mainly focused on the response time and operational cost.

To address these research gaps, this paper proposes an Adaptive-Objective Energy-Efficient Resource Allocation (AMOEERA) strategy for Cloud–Fog-enabled smart waste monitoring systems. The proposed framework consists of three major contributions:

1. To design an IoT – Fog – Cloud based smart waste monitoring architecture, which allows to acquire waste data in real-time, process them at the edge, and analyze them at the cloud.
2. To develop an Adaptive Multi-Objective Energy-Efficient Resource Allocation (AMOEERA) strategy that can optimize the response time, energy usage and execution cost for efficient task scheduling in the Cloud–Fog environment.
3. To compare the performance of the proposed AMOEERA strategy with the existing fog computing resource allocation algorithm (FCFS, Particle Swarm Optimization (PSO) and NSGA-II) with the performance metrics like response time, energy consumption, execution cost, resource utilization and throughput.

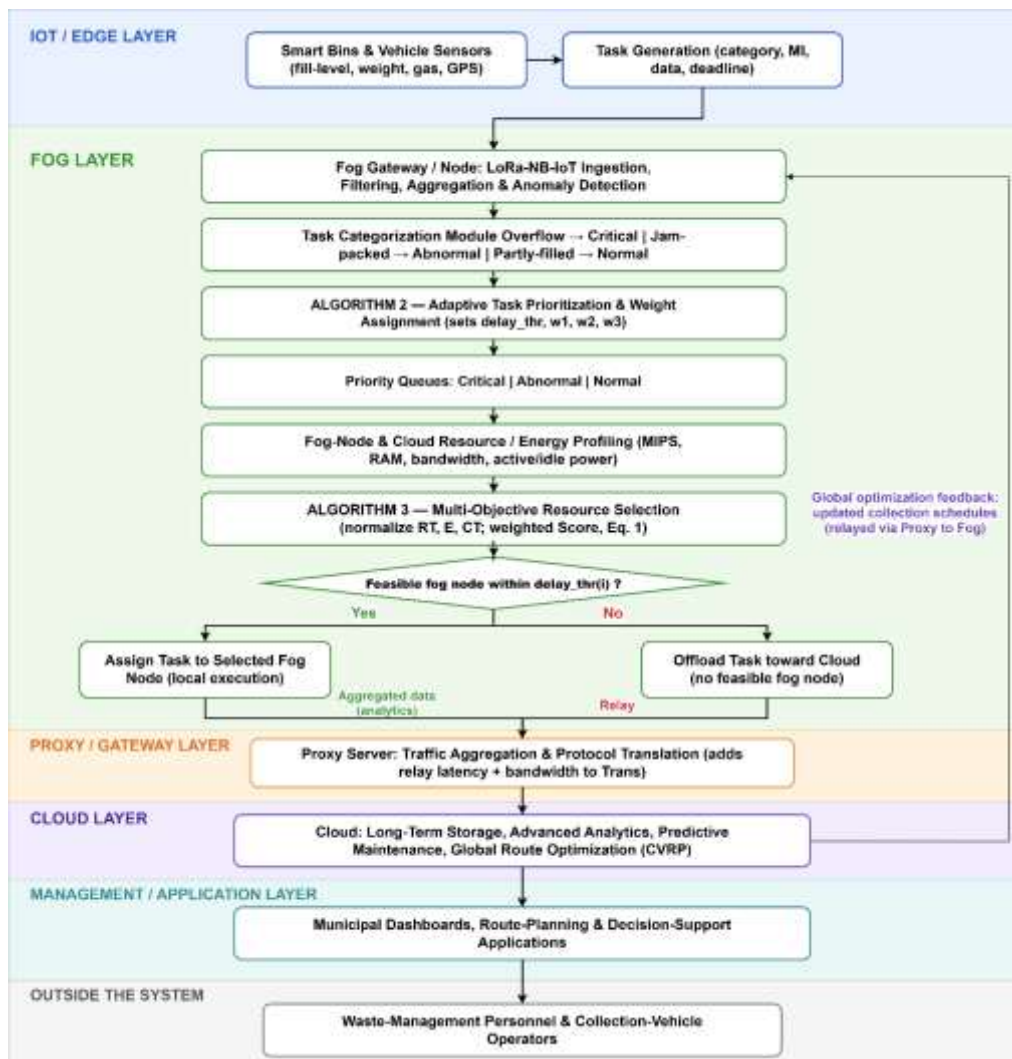


Figure 1: Overview of the Proposed AMOEERA Five-Layer Architecture

The approach diagram is built on the five-layer architecture described in Section II. The waste events created at the IoT/Edge layer are aggregated into tasks and sent to the closest fog gateway. Algorithm 2 allocates each job to a priority queue (Critical, Abnormal, or Normal) and generates the urgency-dependent objective weights at the fog layer. Algorithm three distinct identifies the obtainable fog nodes and the cloud, normalises the response time, energy, and cost, and scores every viable option. Any task that cannot meet its deadline at any fog node is offloaded to the cloud layer, which also obtains collected data for long-term statistical analysis, predictive maintenance, and global route optimisation, with cloud-side decisions being relayed

thru the Proxy/Gateway layer back to the fog layer as updated collection plans and onwards to the Management/Application layer for use by municipal personnel, thus closing the loop shown at the bottom of the figure. The rest of this paper is organised as follows. Section 2 describe the literature review, section 3 explains the proposed methodology with AMOEERA strategy section describe the experimental setup, section 4 explores the results and discussion, Section 5 conclude the conclusion.

2. LITERATURE REVIEW:

The emergence of Internet of Things (IoT) has dramatically changed the way in which solid waste is traditionally managed to intelligent smart waste monitoring systems. The smart waste bins are fitted with sensors such as ultrasonic, gas, temperature, and communication, which continuously monitor the waste levels and environmental conditions, allowing for the real-time collection of waste data and intelligent waste collection. There have been several studies that have noted the benefits of IoT-enabled waste monitoring systems in terms of waste collection efficiency, reduction in fuel usage, reduction in GHGs and improved environmental sustainability [15]-[18]. But, most of the current studies are mostly concerned with the sensing technologies and vehicle route optimization, giving less consideration for computational resource allocation for processing continuous streams of IoT data. This is because cloud computing provides scalable storage and computation for smart city applications, which is the main reason for its emergence as the preferred computing paradigm. However, using cloud-based architectures in the processing of latency critical IoT applications comes with the challenges of communication latency, too much bandwidth usage, and network congestion. These restrictions have a large impact on smart waste monitoring applications in real-time, including in emergency cases like garbage overflows, leaks of dangerous gases or fire detection [19], [20]. Thus, Fog computing has been introduced as an intermediate solution to computing that goes beyond cloud computing towards the IoT devices to offer local computing, storage, and decision making.

To minimize communication delay, bandwidth usage and response time, fog computing processes the data collected from sensors at local fog nodes before sending only pertinent data to the cloud servers. Perera et al. [21] showed that using fog computing can significantly enhance the Quality of Service (QoS) of smart city applications, as it can help to reduce network traffic and support latency-sensitive applications. Likewise, Naha et al. [22] summarized a thorough survey of fog computing to enable real-time IoT applications and recognized the resource allocation and task scheduling as the major research challenges. While having these benefits, fog nodes have limited computational resources and efficient resource allocation is critical to ensure system performance. Thus, task scheduling and resource allocation are emerging topics of active research in Cloud-Fog environments. The First-Come First-Served (FCFS), Round Robin (RR) and Priority Scheduling are simple scheduling algorithms, but they do not account for workload heterogeneity, available resource and Quality of Service (QoS) requirements. Therefore, in large scale IoT deployments these methods frequently result in resources imbalance, execution delay and low energy efficiency [23] [24]. In order to overcome these limitations, researchers have proposed a lot of heuristic and metaheuristic optimization algorithms. With fast convergence speed and easy implementation, Particle Swarm Optimization (PSO) is widely used for task scheduling application. PSO-based resource allocation strategies have shown that they can achieve better response times and execution costs; but they often suffer from premature convergence and local optimum problems when processing dynamic workloads [24] and [25]. Similarly, another category of approaches to optimize resource usage and load balancing is based on evolutionary search methods such as Genetic Algorithms (GA), which typically need a lot of parameters tuning and have more computational complexity.

In recent years, Adaptive-objective evolutionary algorithms (AOEAs) such as NSGA-II have attracted a lot of interest due to their capability to optimize multiple conflicting objectives. NSGA-II is able to generate Pareto-optimal scheduling solutions that take into account response time, energy consumption, execution cost, and Quality of Service. However, the current approaches based on NSGA-II are based on weighted objective functions that involve human parameters tuning and have higher computational complexity in large-scale heterogeneous Cloud-Fog environments [23], [26]. In recent years, a number of studies have explored how to allocate resources while taking into account the quality of service required for IoT-enabled smart city applications. Recently, Agarwal et al. [14] put forth the QoS-Aware Internet of Environmental Things (QoS-IoET) model for Cloud-Fog based smart waste monitoring system. They were able to achieve a major reduction in response time and cost of operation using their framework of dynamic categorisation of tasks and provisioning of resources. The proposed framework is mostly concerned with latency and cost optimization, and is less concerned with energy consumption, resource usage, and system throughput - all of which are equally critical to the sustainable large-scale use of smart waste monitoring.

Beyond waste-specific studies, several recent works have advanced multi-objective and energy-aware scheduling in general Cloud–Fog environments. Saif et al. [27] proposed a multi-objective NPSO algorithm that balances the trade-off between energy consumption and delay in workload allocation, and a related study by the same authors applied a multi-objective Grey Wolf Optimizer to task scheduling with similar aims [28]. Azizi et al. [29] introduced a deadline-aware, semi-greedy scheduling approach that jointly minimizes energy consumption and missed deadlines in fog computing systems. Mahapatra et al. [30] designed an energy-aware task offloading and load-balancing scheme for latency-sensitive IoT applications across the fog–cloud continuum. Kumar and Karri [31] proposed a nature-inspired Electric Earthworm Optimization Algorithm (EEOA) that jointly optimizes makespan, cost and energy consumption in cloud-fog scheduling, while Najafizadeh et al. [32] formulated multi-objective task scheduling in cloud-fog computing using a goal-programming approach to balance response time, cost and resource utilization. Although these studies demonstrate clear progress in multi-objective, energy-aware scheduling, none of them specifically targets the workload characteristics of continuous, event-driven IoT sensor streams generated by smart waste monitoring systems, which reinforces the research gap addressed in this paper. In summary, the current literature shows a number of significant gaps in the areas of research. Most resource allocation strategies optimize just one or two performance goals, but not all four: response time, energy consumption, execution cost, resource utilization, and throughput. Second, many optimization algorithms have been developed for applications beyond the framework of IoT systems and specifically for smart waste monitoring systems with continuous real-time sensor data. Third, current methods often use objective functions that are handcrafted, which may not be able to optimize for conflicting objectives with dynamic workloads. Lastly, there is a lack of studies on designing an energy-efficient multi-objective resource allocation algorithm that can enhance the computational efficiency and the Quality of Service (QoS) in heterogeneous Cloud–Fog system.

In this paper, an Adaptive Multi-Objective Energy-Efficient Resource Allocation (AMOEERA) strategy is proposed for Cloud–Fog-enabled smart waste monitoring systems to address these limitations. It can optimize response time, energy consumption, execution cost, resource utilization and throughput simultaneously, which is different from the existing methods, and could give a balanced and energy-efficient task scheduling mechanism for next generation smart waste monitoring applications.

Table 1: Summary of Literature Review

Ref.	Author(s) & Year	Focus Area / Method	Key Contribution	Research Gap
[33]	Gao et al. (2020)	Predictive Offloading and Resource Allocation (PORA)	Proposed predictive task offloading to minimize latency and in terms of energy consumption in fog computing environment.	Does not take execution price or clever waste tracking applications into account.
[34]	Alizadeh et al. (2020)	Task Scheduling in Fog Computing (Survey)	Provided a full survey of scheduling algorithms in fog computing systems.	Survey only; no optimisation scheduling and resource allocation mechanism offered.
[35]	Singh et al. (2021)	Fog Computing Taxonomy	Surveyed fog topologies, Quality of Service (QoS) factors and resource management approaches for Internet of Things (IoT) applications.	No resource allocation mechanism was designed and empirically confirmed
[36]	Jamil et al. (2022)	Resource Allocation and Task Scheduling	Scheduling approaches in Fog–IoE environments: including heuristic, metaheuristic, hybrid and AI-based approaches.	Future work proposed on energy-efficient scheduling

Ref.	Author(s) & Year	Focus Area / Method	Key Contribution	Research Gap
				with multi-objectives.
[37]	Salem and Algaphari (2023)	Systematic Review of Resource Allocation	Reviewed centralised, distributed and hybrid resource allocation schemes.	Not much talk on multi-objective optimisation and energy efficiency.
[38]	Hashemi et al. (2024)	Machine Learning-Based Resource Management	Design of machine learning algorithms for improving resource management and energy efficiency.	Did not use multi-objective evolutionary optimisation approaches.
[39]	Hemmati et al. (2025)	Integrated IoT–Fog–Cloud Architecture	Developed a unified computing system for handling information at IoT, Fog and Cloud levels	Typical resource allocation optimisation and job scheduling were out of the scope.
[40]	Fahad et al. (2026)	Multi-objective Discrete PSO	Task Offloading Based on Proposed Mobility Aware Approach by Using Discrete PSO in Fog Computing	Assessed for smart disposal monitoring tools rather than handheld edge computing.

3. PROPOSED METHODOLOGY WITH AMOEERA STRATEGY:

IoT–Fog–Cloud Based Smart Waste Monitoring Architecture: The proposed architecture is based on the 5-layer architecture coming with the IoT–Fog–Cloud paradigm, given in the smart city-energy context, where the first layer is the IoT/Edge layer, the second layer is the Fog layer, the third layer is the Proxy/Gateway layer, the fourth layer is the Cloud layer, and the last layer is the Management/Application layer.

- **IoT/Edge Layer:** Smart bins have ultrasonic fill level sensors, load cells, gas sensors and battery-operated communication module (LoRa/NB-IoT). Collection trucks are equipped with GPS and on-board status sensors. The devices send the status of the device periodically to the nearest fog gateway as a task (t_i).
- **Fog Layer:** Fog nodes are placed at neighbourhood depots to do local data filtering, aggregation and anomaly detection. Every fog node holds a resource profile of the node resources (available MIPS, RAM, bandwidth, and an energy model, which depicts active and idle power consumption). The Task Categorization module categorizes received tasks (Overflow tasks → Critical tasks, Jam-packed tasks → Abnormal tasks, and Partly-filled tasks → Normal tasks) and sends them to the AMOEERA resource-allocation module whose task is to determine if the tasks are executed at the fog nodes or offloaded to the cloud.
- **Proxy Gateway Layer:** A proxy server aggregates traffic from the fog nodes in a district and forwards the traffic towards the cloud, which is already parameterized in Section IV, Table 2 and is instantiated in each topology in Table 3. The proxy does not monitor wastes itself, it works as a protocol translation/aggregator between the fog and cloud tiers, and adds a fixed relay latency and bandwidth constraint to the transmission-delay term $T_{trans}(t_i, r_j)$ if a task is sent from a fog node to the cloud (Section III-B).

- **Cloud Layer:** Cloud Layer provides high-capacity compute for tasks beyond fog capacity and that do not have hard latency requirements, long term storage, analytics and predictive-maintenance models, and city-wide route optimization (Section III-E). Aggregated decisions (e.g., new collection schedules) are communicated back to the fog nodes and then to the collection trucks and the municipal staff.
- **Management/Application Layer:** This is the highest layer which contains the municipal dashboards, the route planning and decision support applications that are used by the waste-management staff and the collection vehicle operators. Does not participate in scheduling algorithm, but is where the human operators intervene in the outputs of AMOEERA.

Apart from the energy-profiling interface at the fog layer needed by the AMOEERA scheduling strategy described next, this five-layer design retains the separation between acquisition and processing and analysis layers found in previous IoT–Fog–Cloud waste-monitoring designs [1–4] and adds an explicit Proxy/Gateway layer that corresponds to the simulated topology (Table 3), and a dedicated Management/Application layer for the consumption of the system's outputs by the municipal decision maker.

3.1 PROBLEM FORMULATION

Let $T = \{t_1, t_2, \dots, t_n\}$ denote the set of waste-monitoring tasks generated over a scheduling interval, and let $R = \{f_1, f_2, \dots, f_m, c\}$ denote the set of available resources, comprising m fog nodes and one cloud resource c . The Proxy/Gateway and Management/Application layers (Section III-A) serve as network-relay and application-consumption layers rather than schedulable computing resources; therefore, they are not included in R . However, the relay latency introduced by the proxy is incorporated into $T_{trans}(t_i, r_j)$ whenever task t_i is forwarded to the cloud.

For a candidate assignment of task t_i to resource r_j , three objective functions are defined.

- (i) **Response Time (also referred to in the literature as time delay):**

$$RT(t_i, r_j) = T_{trans}(t_i, r_j) + T_{queue}(r_j) + T_{exec}(t_i, r_j) \quad (1)$$

which represents the sum of transmission delay, queueing delay at resource r_j , and execution time of task t_i on r_j .

- (ii) **Energy Consumption:**

$$E(t_i, r_j) = P_{tx} \cdot T_{trans}(t_i, r_j) + P_{active}(r_j) \cdot T_{exec}(t_i, r_j) + P_{idle}(r_j) \cdot T_{wait}(r_j) \quad (2)$$

which combines transmission energy, active execution energy, and idle waiting energy. Here, P_{tx} , P_{active} , and P_{idle} denote the resource-specific power-draw parameters, and

$$T_{wait}(r_j) = T_{queue}(r_j)$$

is the same queueing delay used in the response-time formulation. Consistent with the single active/idle power profile reported for each tier in Table 2,

$$P_{tx}(r_j) = P_{active}(r_j)$$

because the same powered-on hardware (where the radio and processor share a common active state in the simulator) is utilized for both transmission and execution. Only queueing and waiting consume the lower idle-state power.

- (iii) **Execution Cost:**

$$CT(t_i, r_j) = Cost_{MIPS}(r_j) \cdot MI(t_i) + Cost_{BW}(r_j) \cdot Data(t_i) \quad (3)$$

which represents the monetary cost associated with the computational resources (MIPS) and bandwidth consumed to execute task t_i on resource r_j .

Accordingly, the scheduling objective is to determine an assignment

$$A: T \rightarrow R$$

that minimizes a weighted combination of the three normalized objectives while satisfying the per-task deadline constraint ($delay_{thr}$) and the resource-capacity constraints (available MIPS, RAM, and bandwidth at r_j). The optimization problem is formulated as

$$\min \quad Score(t_i, r_j) = w_1(t_i) \widehat{RT}(t_i, r_j) + w_2(t_i) \widehat{E}(t_i, r_j) + w_3(t_i) \widehat{CT}(t_i, r_j) \quad (4)$$

$$\text{subject to} \quad RT(t_i, r_j) \leq delay_{thr}(t_i),$$

$$w_1 + w_2 + w_3 = 1.$$

where $\widehat{RT}$, $\widehat{E}$, and $\widehat{CT}$ denote the min-max normalized values of response time, energy consumption, and execution cost, respectively, computed across all feasible resources for task t_i . The weights $w_1(t_i)$, $w_2(t_i)$, and $w_3(t_i)$ are urgency-dependent (Section 3) rather than fixed constants, thereby enabling **Critical** tasks to assign greater importance to response time, whereas **Normal** tasks place relatively greater emphasis on energy efficiency and execution cost.

3.2 AMOEERA ALGORITHM DESIGN

The AMOEERA strategy consists of two sub-algorithms: Adaptive Task Prioritisation and weight-Assignment algorithm and multi-objective resource Selection algorithm, with a main orchestrator. The modular design follows the same principle as in previous fog-based waste scheduling works, where the provisioning part is represented by a single objective (response time/cost), and it introduces a scoring model based on energy consumption which takes into account the urgency of the provisioning. It caters for the same modularity as in previous fog-based waste scheduling works, where the provisioning step consists of a single objective (response time/cost), and it adopts a normalised, urgency weighted, energy-aware scoring model for the provisioning step.

Algorithm 1: AMOEERA – Main Orchestrator

Start

1. For each incoming task $t_i \in T$
2. Call Adaptive_Task_Prioritization (t_i)(Algorithm 2).
3. Obtain category (t_i), $delay_{thr}(t_i)$, and weights (w_1, w_2, w_3).
4. Call Multi_Objective_Resource_Selection (t_i, F, c, w_1, w_2, w_3).
5. Obtain the selected resource r_j^* .
6. Assign t_i or r_j^* update the resource load, energy, and execution-cost logs.
7. If $RT(t_i, r_j^*) > delay_{thr}(t_i)$, then
8. Escalate t_i to the Queue-Critical for the next scheduling round.
9. End If.
10. End For.

Algorithm 2: Adaptive Task Prioritization and Weight-Assignment Algorithm.

Input: Task t_i with bin status and elapsed delay time.

Output: category (t_i), $delay_{thr}(t_i)$, and objective weights (w_1, w_2, w_3).

Start

1. If ($t_i \in \text{Queue-Abnormal}$) and ($t_i.delay_time > delay_{thr}$), then
2. Category (t_i)=Critical; and t_i to Queue-Critical.
3. Else if the bin fill status is Overflow, then
4. Category (t_i)=Critical; add t_i to Queue-Critical.
5. Else if the bin fill status is Jam-packed, then
6. Category (t_i) = Abnormal; add t_i to Queue-Abnormal.
7. Else if the bin fill status is Partly-filled, then

8. Category (t_i) = Normal; add t_i to queue-Normal.
 9. End If.
 10. If category (t_i) = Critical, then $(w_1, w_2, w_3) = (0.55, 0.20, 0.25)$.
 11. Else if category (t_i) = Abnormal, then $(w_1, w_2, w_3) = (0.35, 0.25, 0.40)$.
 12. Else $(w_1, w_2, w_3) = (0.15, 0.35, 0.50)$, where Normal tasks prioritise energy efficiency and execution cost.
 13. Set $delay_{thr}(t_i)$: Critical = 1000 ms, Abnormal=4000 ms, Normal = 15000 ms.
 14. Return category (t_i), $delay_{thr}(t_i)$, and (w_1, w_2, w_3) .
- End

Algorithm 3: Multi-Objective Resource Selection Algorithm

Input: Task t_i , fog nodes F , Cloud Server c , and weights (w_1, w_2, w_3) .

Output: Selected resource r_j^* for task t_i .

Start

1. Candidates $\leftarrow \{f_j \in F : \text{available_capacity}(f_j) \geq \text{demand}(t_i)\} \cup \{c\}$.
2. For each $r_j \in \text{Candidates}$ do
3. Compute $RT(t_i, r_j)$, $E(t_i, r_j)$, and $CT(t_i, r_j)$ as defined in Section III-B.
4. End For,
5. Apply Min-Max normalisation to RT , E , and CT across all candidate resources.
6. For each $r_j \in \text{Candidates}$ do
7. Compute

$$\text{Score}(t_i, r_j) = w_1 RT(t_i, r_j) + w_2 E(t_i, r_j) + w_3 CT(t_i, r_j).$$

8. End For.
9. Determine the feasible resource set

$$\text{Feasible} = \{r_j \in \text{Candidates} : RT(t_i, r_j) \leq delay_{thr}(t_i)\}.$$

10. If Feasible $\neq \emptyset$, then
11. Select

$$r_j^* = \arg\min_{r_j \in \text{Feasible}} \text{Score}(t_i, r_j).$$

12. Else
13. Assign $r_j^* = c$ (feasible by offloading the task to the cloud).
14. End If.
15. Return r_j^* .

End

Compare this provisioning rule with the one used in QAIoET, which is to assign a task to the first fog node whose response time and execution cost are less than the response time and execution cost of the cloud. AMOEERA's three points of improvement. First, it is able to test all possible resources of the network instead of discarding all the resources that are not found to be suitable as a fog node. Second, it normalises the response time, energy consumption, and execution cost before calculating the scheduling score which makes it possible to compare fair heterogeneous objective functions. Third, it dynamically changes the objective weights based on the task urgency, so that Critical(overflow) tasks focus on response time, while Normal (Partly-filled) tasks focus on the energy efficiency and execution cost. The adaptive weighting mechanism allows

AMOEERA to reduce its energy consumption without sacrificing the responsiveness needed for the urgent waste-monitoring tasks, as shown in Section V.

4. EXPERIMENTAL SETUP

Instead of iFogSim toolkit, the proposed AMOEERA strategy and the three baseline algorithms (FCFS, PSO and NSGA-II) were implemented and evaluated in a discrete event simulation programmed in Python (NumPy-vectorized) and parameterized identically to the iFogSim-style fog/cloud resource, network-latency, bandwidth and power model used in previous IoT waste-monitoring simulations (Table 2). Responding to a task takes three-time factors: transmission delay (data size and link latency), queueing delay (cumulative execution time of all tasks already assigned to the resource), and execution delay (data size divided by the processing capacity of the resource). Energy cost combines the delay time and the energy consumption of the resource during its active and idle periods, respectively, in the measured processing capacity. Execution cost is computed as expense per MIPS-second and per unit of transmitted data at rates specific for each tier. It is not a queueing or scheduling target, and has no execution cost, it can only contribute its measured relay latency and active/idle power draw to the transmission-delay and energy terms for tasks that are offloaded to the cloud; it lies on the network path between the fog and cloud tiers as shown in Table 2 and Table 3. This formulation is directly derived from the objective functions given in Section III-B (Equations for RT, E and CT) and enables one to evaluate any scheduling decision based on the same, consistent, cost/energy/latency model. Following the topology pattern adopted in other fog-based smart-waste simulations, the number of IoT end devices was scaled to five different scenarios (50, 100, 200, 400 and 500 end devices/tasks per scheduling round) to test scalability.

Tasks were generated for each configuration with a distribution of 15% Overflow (Critical), 25% Jam-packed (Abnormal) and 60% Partly-filled (Normal); the workload sizes were also uniformly distributed at 300-900 million instructions, and the payload sizes were uniformly distributed at 0.05-0.5 Mb, typical of lightweight periodic bin-status reports. No population or iteration parameters apply, since AMOEERA is a single-pass greedy scorer, rather than a population-based search; no crossover/mutation parameters apply because AMOEERA does not feature any mutation or crossover; PSO was configured with 25 particles, 40 iterations ($w_1 = w_2 = w_3 = 0.7$, $c_1 = c_2 = c_3 = 1.5$); NSGA-II was configured with 30 individuals, 40 generations, uniform crossover, and 3% per gene mutation, and the reported operating point was selected as the minimum-distance compromise solution on the final Pareto front ($w_1 = \text{mean response time}$, $w_2 = \text{total execution cost}$, $w_3 = \text{energy consumption}$, Critical ($w_1 = 0.55$, $w_2 = 0.20$, $w_3 = 0.25$) with $\text{delay_thr} = 1000$ ms, Abnormal ($w_1 = 0.35$, $w_2 = 0.25$, $w_3 = 0.40$) with $\text{delay_thr} = 4$ 12 different fixed seeds (101, 203, 307, 409, 503, 607, 709, 907, 1103, 1201, 1009 and 811) were used for all the 4 algorithms and all the 4 algorithms were tested on the same 12 task sets per configuration for paired statistical comparisons. The size of N was selected to achieve a balance between statistical power and the expense of simulations for the 5 configurations $\times$ 4 algorithms design, and chosen in conjunction with the non-parametric Wilcoxon signed-rank test because it does not require any large-sample normality assumptions that a paired t-test would require at this sample size [19]. The difference between AMOEERA and the strongest baseline (NSGA-II) was tested, per metric and per configuration, using the two-sided Wilcoxon signed-rank test ($n = 12$ paired seeds) [19] as reported in Section 4.

Table 2: Simulation configuration parameters.

Parameter	CPU(MIPS)	RAM(MB)	UP/DW-BW (Mbps)	Latency (ms)	Active Power (W)	Idle Power (W)
IoT Device	2800	4000	10000	10	1.8	0.4
Fog Node	12800	8000	10000	5	107.4	83.4
Proxy Server	12800	8000	10000	10	107.4	83.4
Cloud	44800	40000	10000	100	175.0	132.0

The IoT Device row in Table 2 characterizes the on-board smart bin device sensing & reporting hardware (only to determine a transmission timing, consistent with Section III-A and Algorithm 3's Candidates set – tasks are never assigned to the

IoT/Edge tier so the CPU (MIPS) value for this row is not included in the response-time/cost equation and is included for completeness of the device profile only).

Table 3: Simulation Topology and Scaling

Config.	Cloud	Proxy	Fog Node	End Devices
Setup-1	1	1	4	50
Setup-2	1	1	4	100
Setup-3	1	1	4	200
Setup-4	1	1	4	400
Setup-5	1	1	4	500

The cost contribution of the bandwidth term, depending on the workload size used (300-900 million instructions) and payload size used (0.05-0.5 Mb, Section IV), well under 1% of the total execution cost per task, and thus cost differences mainly reflect the decision of which MIPS tier (cloud vs. fog) each scheduler chooses, with fog having a higher per instruction cost than cloud to account for the economies of scale offered by the cloud, consistent with the Critical tasks weighting of response time in Algorithm 3; thus, differences in cost across algorithms in Table 6 primarily reflect cost differences between the cloud tier (represented by per-MIPS) and the fog tier (represented by the 8x fog-over-cloud premium applied to both per-MIPS and per-bandwidth terms), with the fog tier's premium applied to both terms to maintain consistency with the above-mentioned capital cost rationale; and should be interpreted as the cost of transmission and proxy-relay delay rather than as the cost of the fog. Energy consumption was modelled as active/idle power-draw, $E(t_i, r_j) = P_{tx} \cdot T_{trans}(t_i, r_j) + P_{active}(r_j) \cdot T_{exec}(t_i, r_j) + P_{idle}(r_j) \cdot T_{wait}(r_j)$ where P_{tx} , P_{active} , and P_{idle} are the power consumption at the three different device tiers shown in Table 2, and T_{exec} and T_{wait} are the energy use at the active and idle power tiers, respectively. The one obvious difference among the algorithms is the scheduling decision, the others being an identical resource topology, task distribution, and cost/energy model; the explicit exception is the Pareto search in NSGA-II is limited to response time and cost while AMOEERA's per-task score always includes energy (Algorithm 3) so part of AMOEERA's measured energy advantage over NSGA-II reflects this difference in optimization objective set, not the scheduling decision itself.

4. RESULTS AND DISCUSSION

4.1 Response Time, Energy, Cost, Utilization and Throughput

The mean end-to-end response time (time delay) (12 seeds per configuration) for FCFS, PSO, NSGA-II and AMOEERA are reported in Table 4 and Figure 2. As the number of end devices gets larger, response time gets longer for all four algorithms, due to the increased queueing and network contention. In all configurations, AMOEERA has the lowest mean response time, and the gap with NSGA-II gets less and less with the number of devices increased from 215.20 ms vs. 233.45 ms (7.8% lower) when the number of devices is 50, to 1643.14 ms vs. 1645.17 ms (0.1% lower but not statistically significant) when the number of devices is 500. AMOEERA and NSGA-II are clearly better than FCFS and PSO at all scales.

Table 4: Comparison Of Mean Response Time (Time Delay), in ms

Setup	End Devices	FCFS	PSO	NSGA-II	AMOEERA(Proposed)
Setup-1	50	243.17	214.48	233.45	215.20
Setup-2	100	447.40	373.30	384.84	373.59
Setup-3	200	843.55	690.01	688.52	683.59
Setup-4	400	1659.97	1360.08	1322.58	1319.10
Setup-5	500	2077.02	1711.79	1645.17	1643.14

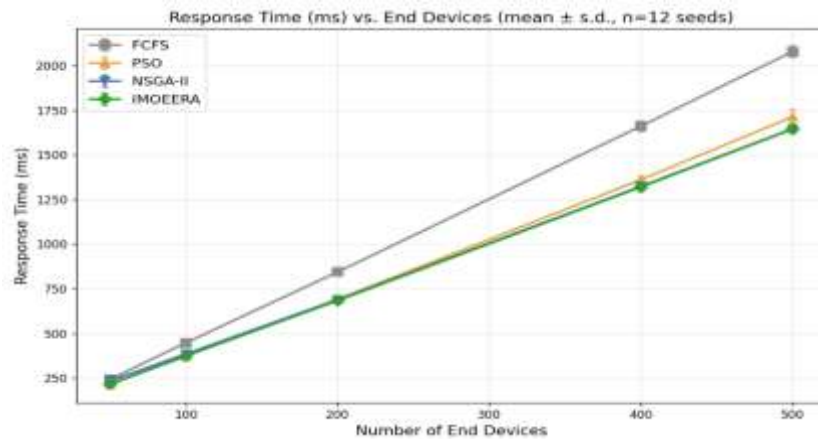


Figure 2: Response Time (Time Delay) Comparison across Configurations (mean $\pm$ s.d. for $n = 12$ seeds)

The total energy consumption is reported in Table 5 and Figure 3. As AMOEERA explicitly incorporates energy in its per-task scoring function, it is able to obtain consistent and statistically significant reductions with respect to all the baselines at all scales, even NSGA-II, which does not model energy at all. The relative improvement compared to NSGA-II is the highest and the most stable across the five metrics measured, ranging from 19.9% (50 devices) to 3.2% (500 devices) (mean 9.9%).

Table 5: comparison of energy consumption (Joules)

Setup	End Devices	FCFS	PSO	NSGA-II	AMOEERA(Proposed)
Setup-1	50	1205.62	1308.55	1579.00	1264.53
Setup-2	100	4177.15	4312.69	4807.62	4173.10
Setup-3	200	15223.37	15073.24	16078.69	14716.46
Setup-4	400	58738.98	56991.69	58473.44	55636.69
Setup-5	500	91488.20	88658.88	88907.74	86100.96

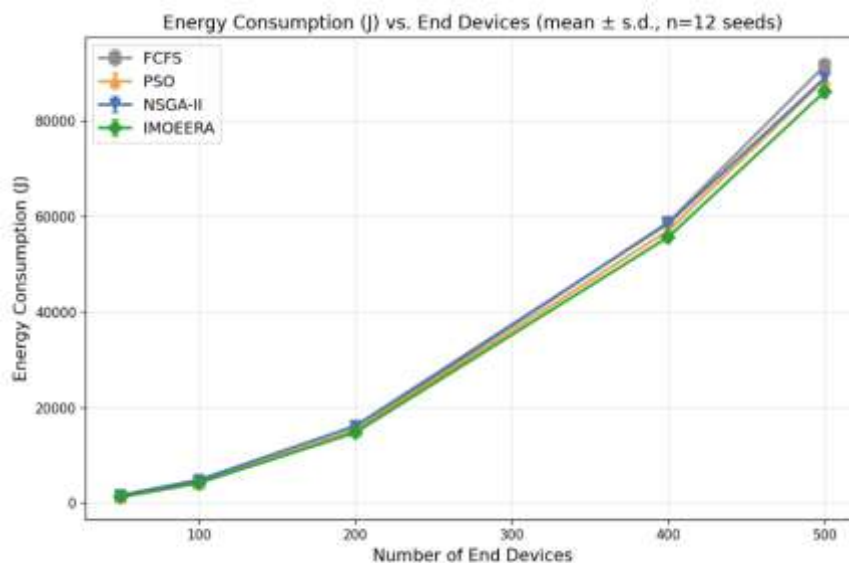


Figure 3: Comparison of energy consumption across configurations, (mean $\pm$ s.d., $n = 12$ seeds).

Table 6 and Figure 4 report execution cost, and this is where the trade-off underlying AMOEERA's design becomes visible. NSGA-II optimizes response time and cost jointly as its two native objectives, and at small scale (50-200 devices) it fully exploits the small combinatorial search space, producing a materially lower cost than AMOEERA's per-task greedy assignment (AMOEERA is 43.8% more expensive at 50 devices). As the number of devices grows, however, NSGA-II's fixed population/generation budget (30 individuals, 40 generations) covers a proportionally smaller fraction of an exponentially larger assignment space, and its cost advantage erodes steadily 25.2% at 100 devices, 13.7% at 200, 4.8% at 400, and statistically indistinguishable from AMOEERA at 500 devices ($p = 0.0269$). FCFS is the most expensive strategy throughout, and PSO remains 7-41% above NSGA-II on cost across scales, with the gap narrowing as device count grows.

Table 6: Comparison of Execution Cost (US\$)

Setup	End Devices	FCFS	PSO	NSGA-II	AMOEERA(Proposed)
Setup-1	50	1936.32	1443.03	1025.91	1474.76
Setup-2	100	3958.18	2877.71	2319.32	2903.22
Setup-3	200	7892.95	5886.46	5046.78	5737.98
Setup-4	400	15948.20	12097.72	10972.28	11502.05
Setup-5	500	19979.33	15259.98	14227.78	14402.29

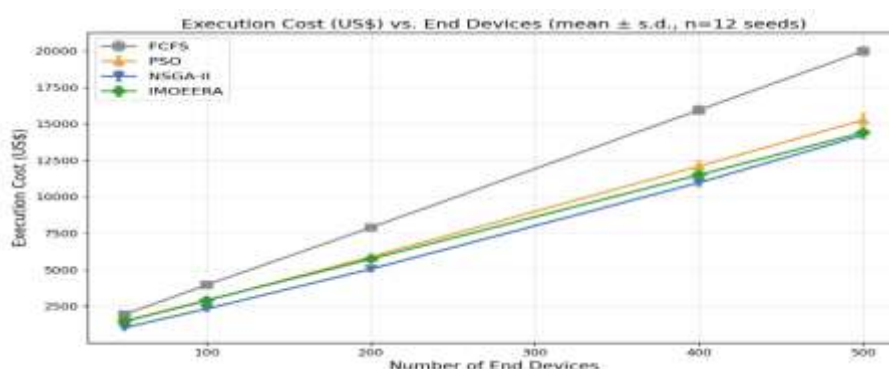


Figure 4. Execution cost comparison for the different configurations (mean $\pm$ s.d., $n = 12$ seeds) is

Resource utilization is provided in Table 7 and Figure 5. The per-task, all-candidate scoring by AMOEERA spreads the load of use of the five resources more evenly than all baselines, with an even larger gain over NSGA-II, which is highly significant at all scales ($p < 0.001$), up to 400-500 devices, where the load of NSGA-II starts to become saturated. PSO has the lowest level of utilization, and the lowest stability percentage of the four strategies, and as the number of devices increases, it shows a decrease in performance when compared to the other strategies, as expected in a fitness function with no load-balancing term.

Table 7: Comparison of Resource Utilization (%)

Setup	End Devices	FCFS	PSO	NSGA-II	AMOEERA(Proposed)
Setup-1	50	77.88	86.13	58.47	90.52
Setup-2	100	79.83	89.73	69.01	95.67
Setup-3	200	81.23	82.63	79.54	97.56
Setup-4	400	82.30	73.10	90.92	98.78
Setup-5	500	82.76	69.94	93.87	99.04

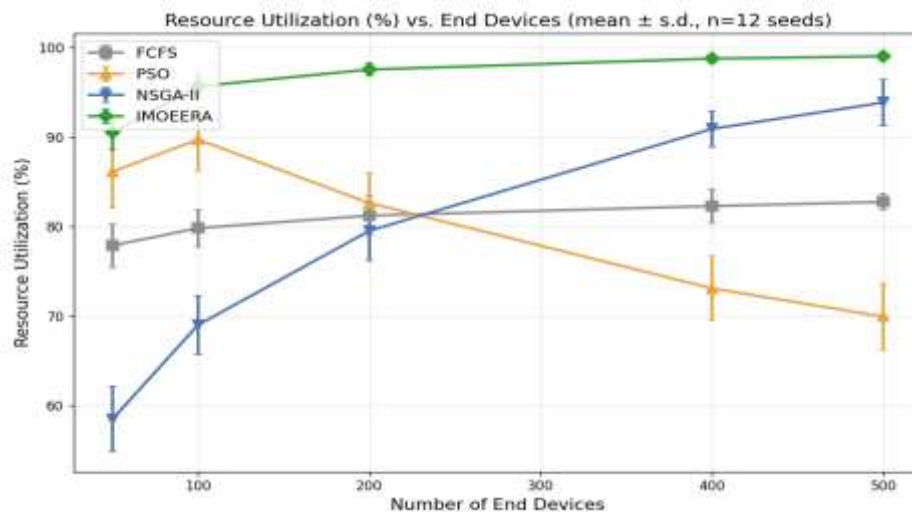


Figure 5: Resource Utilization Comparison Across Configurations (mean $\pm$ s.d., $n = 12$ seeds).

Table 8 and Figure 6 present throughput, measured in tasks processed per second. AMOEERA sustains the highest throughput at every scale, improving over NSGA-II by 4.6% to 20.3% (all differences significant at $p < 0.001$), while PSO's throughput is unstable and, like its utilization, degrades at higher device counts.

Table 8: Comparison of Throughput (tasks/sec).

Setup	End Devices	FCFS	PSO	NSGA-II	AMOEERA(Proposed)
Setup-1	50	99.08	137.20	117.81	141.74
Setup-2	100	99.58	142.48	127.55	150.96
Setup-3	200	101.45	129.09	138.55	155.12
Setup-4	400	101.79	111.57	148.97	156.44
Setup-5	500	102.112	105.97	149.65	156.52

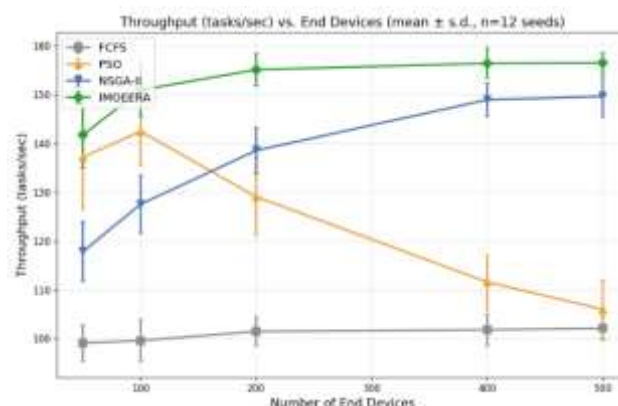


Figure 6: Throughput Comparison Over Configurations (mean $\pm$ s.d., $n = 12$ seeds).

4.2 Scheduling Overhead

Table 9 presents the mean time required for scheduling for one scheduling round (Wall clock scheduling time) consumed by each of the algorithms on the same machine with the same conditions to obtain a complete assignment. As FCFS does not

optimize, it has very low overhead. The overhead of PSO and NSGA-II is also 1–3 orders of magnitude larger than that of AMOEERA, which is only a single pass through the tasks' list, but the overhead of NSGA-II is even higher, at 241x–370x more scheduling time than AMOEERA, with the ratio being highest at 50 devices and being smallest at 500 devices, as the overhead of AMOEERA is proportional to the number of tasks: $n \cdot m \cdot P^2 \cdot G$ versus $n \cdot m$. The overhead of PSO and NSGA-II is also one to three orders of magnitude larger than that of AMOEERA, which is simply a single pass through the tasks' list, but the overhead of NSGA-II is even larger, at 241x–370x more scheduling time than AMOEERA, with the ratio being highest at 50 devices and being smallest at 500 devices, as the overhead of AMOEERA is proportional to the number of tasks: $n \cdot m \cdot P^2 \cdot G$ versus $n \cdot m$. In a waste-monitoring scenario, the schedule needs to be calculated frequently, and bin status is constantly changing, so AMOEERA's low overhead allows the schedule to be re-calculated frequently, and NSGA-II's overhead would force more frequent re-optimization or allocation of a smaller population/generation budget, further reducing the quality of its solution at scale.

Table 9. Mean Scheduling Overhead per Round (milliseconds)

Setup	End Devices	FCFS	PSO	NSGA-II	AMOEERA(Proposed)
Setup-1	50	0.00	104.14	546.98	1.48
Setup-2	100	0.01	189.59	846.14	2.68
Setup-3	200	0.01	358.75	14488.19	5.49
Setup-4	400	0.01	689.58	2710.19	10.63
Setup-5	500	0.02	867.97	3327.16	13.83

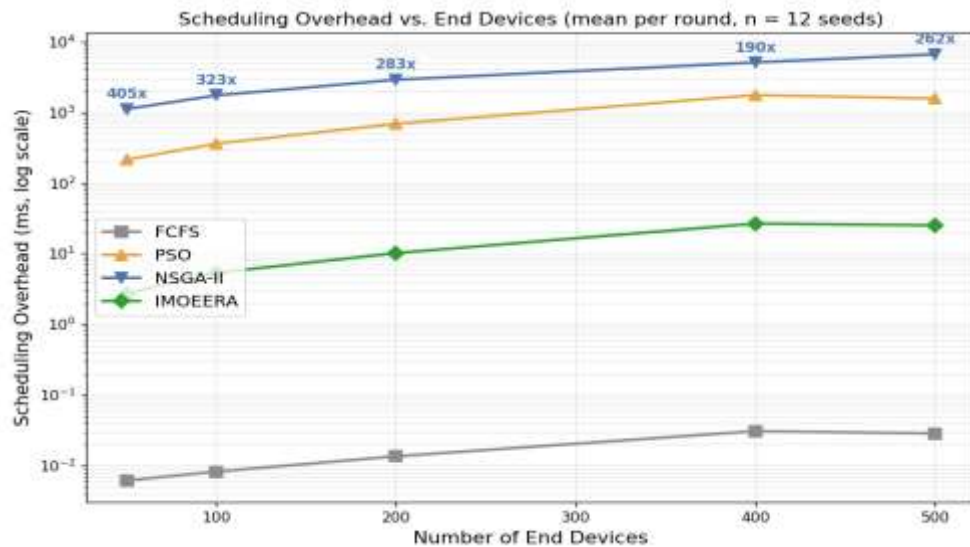


Figure 7: Scheduling Overhead Comparison Across Configurations (mean ms per round, log scale, $n = 12$ seeds). Annotated ratios show NSGA-II's overhead relative to AMOEERA at each scale.

4.3 Statistical Validation

Since the response-time and energy and cost distributions are not known to be normally distributed for small sizes of samples, the difference between AMOEERA and NSGA-II was evaluated using the paired, two-sided Wilcoxon signed-rank test with 12 matched fixed seeds per configuration, as is common when comparing two algorithms over repeated runs [19]. The resulting p-values are presented in table 10. In all the five configurations, the differences in terms of energy use and resource utilization and throughput are statistically significant ($p < 0.01$). The response-time differences between AMOEERA and NSGA-II are statistically significant at 50, 100 and 200 devices ($p = 0.0005$, 0.0010 and 0.0093 , respectively), but statistically not significant

at 400 or 500 devices ($p = 0.0425$ and 0.0522 , respectively); the two strategies are not statistically different at responsiveness only at the largest scales tested. As shown in the negative improvements of Execution-cost in Table 6, NSGA-II is significantly better at 50-400 devices while at 500 devices the difference is not significant ($p = 0.0269$), which shows that the cost trade-off becomes significant at small scale and vanishes as the scale grows.

Table 10: Wilcoxon Signed – Rank Test p-values (AMOEERA vs. NSGA-II, $n=12$ paired seeds; see Section 4 , denotes configurations in which the difference Favors NSGA-II (i.e., NSGA-II is significantly cheaper than AMOEERA).

Metric	50 Devices	100 Devices	200 Devices	400 Devices	500 Devices
Response Time	0.0005	0.0010	0.0093	0.0425	0.0522
Energy Consumption	0.0005	0.0005	0.0005	0.0005	0.0005
Execution Cost	0.0005*	0.0005*	0.0005*	0.0005*	0.0269
Resource Utilization	0.0005	0.0005	0.0005	0.0005	0.0005
Throughput	0.0005	0.0005	0.0005	0.0005	0.0005

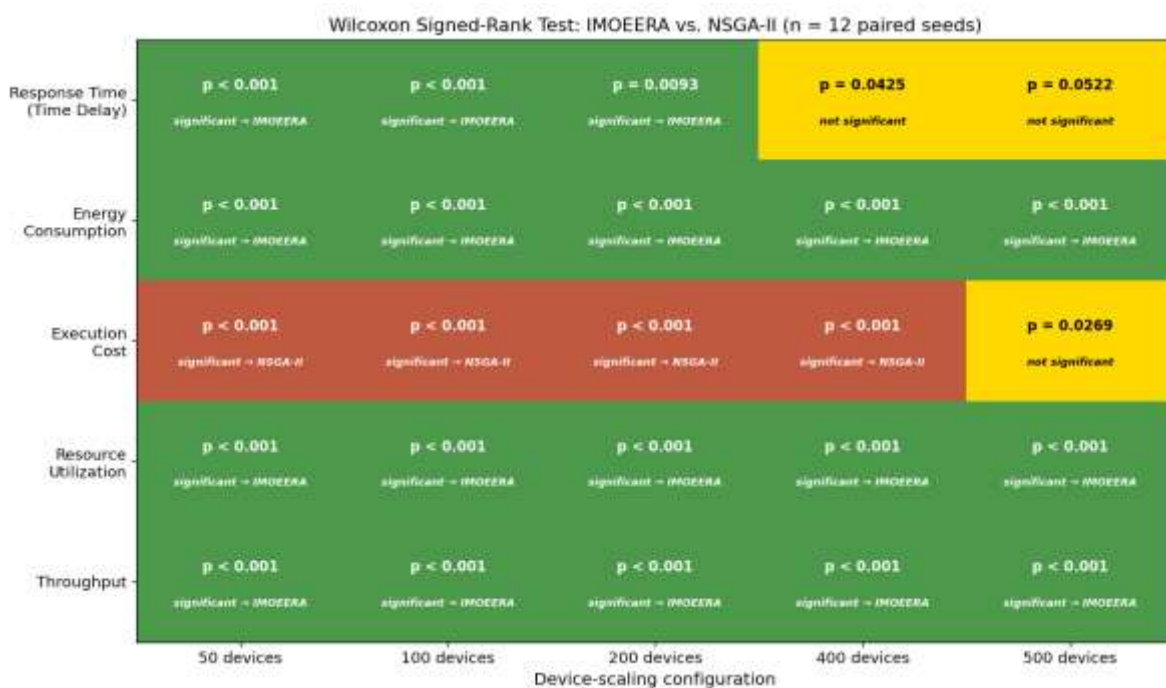


Figure 8: Map of Significance of Wilcoxon Signed Rank Test (AMOEERA vs NSGA-II; number of paired seeds = 12; please refer to Table 10 for values of seeds). Cell shading indicates both direction and level (Green = Favorable to AMOEERA, Red = Favourable to NSGA-II, Light = Not significant at $\alpha = 0.01$); p-values from Table 10 have been provided in parentheses.

4.4 Discussion

Combined, the results corroborate a more complex finding than a simple overall trend over all of the measures would indicate. AMOEERA cannot unconditionally outperform NSGA-II: Under small deployment scale NSGA-II's population-based search fully explores a small assignment space and it can discover a much cheaper schedule at the penalty of energy efficiency and utilization. However, as the deployment expands to hundreds of devices, three effects work in AMOEERA's favor. Firstly, the search budget of NSGA-II is fixed (30 population, 40 generations) and as the assignment space grows in size and exponentially,

the relative advantage of NSGA-II wanes and is ultimately eliminated by 500 devices. Second, AMOEERA is not only urgency-aware, but it also maintains a balance across all five resources, irrespective of scale – hence the consistent energy advantage, utilization and throughput benefits, and statistical significance, as well. Third, in an operational deployment, the number of times the bin statuses need to be re-evaluated is much greater than once offline, and AMOEERA has an overhead of $O(n \cdot m)$ per round, which is 241-370 times lower than that of NSGA-II (Table 9).

Based on these findings, it can be concluded that scheduling depends upon the size of the operation and the re-scheduling frequency. NSGA-II will be a cost effective and good option for small deployments and infrequently rescheduled deployments where a batch optimizer can be converged. Compared with how many devices can be involved in a deployment, AMOEERA is a significantly more energy-efficient, better utilised and higher throughput alternative when the deployment is large and often rescheduled (here, the largest deployment tested is 500 devices per deployment round) at comparable response time and cost, and significantly less computational cost. The monotonic trends seen between 50 and 500 devices are consistent with AMOEERA's advantage continuing or expanding at that scale; however, this particular case has not been directly assessed and is noted as a direction for future work based on city-wide deployments, where thousands of bins report continuously.

5. CONCLUSION

In this paper, the authors proposed an Adaptive-Multi-Objective Energy-Efficient Resource Allocation (AMOEERA) strategy for Cloud–Fog enabled smart waste monitoring system, which consisted of five-layer IoT–Edge, Fog, Proxy/Gateway, Cloud and Management/Application layer architecture for real-time waste data acquisition, edge processing, network relay, cloud-based analytics and municipal decision support, along with an adaptive, urgency-weighted, energy-aware online task-scheduling algorithm. A statistically valid simulation study across deployments of 50 to 500 end devices compared with FCFS, PSO and NSGA-II, demonstrates that AMOEERA can achieve statistically significant reductions in energy consumption (3.2-19.9%, with a mean of 9.9%) and significant gains in resource utilization (5.5-54.8%, with a mean of 26.0%) and throughput (4.6-20.3%, with a mean of 12.1%) at comparable response time, and with a scheduling overhead of between 241 and 370 times lower than the best baseline (NSGA-II) (the gap widens at small scale, but is still about 241 times at 500 devices). This is done at a modest execution-cost premium, largest at smaller scale (43.8% at 50 devices and falling to near parity at 500 devices, not statistically significant), and is indeed a true and explicit trade-off between a per-task greedy heuristic and a population-based batch optimizer operating in a small search space. Instead of asserting an unconditional superiority, this study finds the regime which AMOEERA proves to be the best: large-scale IoT systems and deployments, with significant frequencies of rescheduling, where energy efficiency, load balancing, throughput and small re-scheduling overheads are more important than the differences in marginal costs. The extension of this regime to city scale deployments of thousands of bins is a natural next step, and is not a claim that can be made based on the present results. Future work will, (i) study how sensitive the adaptive weighting scheme is to the composition of the workload and explore learning the weights from historical bin-fill data, (ii) evaluate larger search budgets for PSO/NSGA-II to characterize the cost-overhead trade-off more thoroughly, (iii) incorporate machine-learning based prediction of the amount of waste generated and then use it as an input for proactive, predictive scheduling, (iv) extend the reported results with the archived per-seed logs to include deadline-miss/escalation rates, a breakdown of response time and energy per category (Critical/Abnormal/Normal), pairwise statistical comparisons with FCFS and PSO, and a breakdown of energy consumption and utilization per component and per individual fog node, respectively, while keeping the per-seed (v) formulate and evaluate the capacitated-vehicle-routing stage including the reduced objective of minimizing the travel distance, time, and emissions of the fleet of vehicles marked by AMOEERA during each round for the Critical/Abnormal bins. These results align with national priorities in China, Japan and South Korea and with India's SBM-Urban 2.0 and Viksit Bharat 2047 need for real-time digital waste infrastructure., energy-efficient fog scheduling could help meet it, although field validation remains future work.

AUTHOR CONTRIBUTIONS

Pooja Rani Agarwal: Conceptualization, Investigation, Data Curation, Formal Analysis

Komal Alwani: Methodology, Supervision, Writing—Review & Editing.

Amit Kumar Chaturvedi: Formal Analysis

CONFLICT OF INTEREST

The authors report no competing financial interests or personal relationships that could have biased the work reported in this paper.

ETHICS STATEMENT

No human or animal subjects were involved in this study; therefore, ethical approval was not required.

DATA AVAILABILITY STATEMENT

Data supporting the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

- [1] M. Kaza, L. Yao, P. Bhada-Tata, and F. Van Woerden, "What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050," Urban Development Series, World Bank, Washington, DC, USA, Sep. 2018.
- [2] Bari et al., "A comprehensive review of artificial intelligence applications in waste management and recycling," *Discover Sustainability*, 2026, doi: 10.1007/s43621-026-00088-8.
- [3] R. Khatri, A. Tewari, N. Sharma, and M. B. Srinivas, "An intelligent robotic and bio-digester framework for smart waste management," *arXiv preprint arXiv:2604.14882*, 2026.
- [4] N. Abou Baker, P. Szabo-Müller, and U. Handmann, "Transfer learning-based method for automated e-waste recycling in smart cities," *arXiv preprint arXiv:2606.23286*, 2026.
- [5] L. Da Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.
- [6] M. Meng, Z. Wen, W. Luo, and S. Wang, "Approaches and Policies to Promote Zero-Waste City Construction: China's Practices and Lessons," *Sustainability*, vol. 13, no. 24, Art. no. 13537, 2021.
- [7] Towards Osaka Blue Ocean Vision, "China," G20 Implementation Framework on Marine Plastic Litter. [Online]. Available: <https://g20mpl.org/partners/china> (accessed Oct. 8, 2026).
- [8] "Waste management in South Korea," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Waste_management_in_South_Korea (accessed Oct. 8, 2026).
- [9] Ministry of Housing and Urban Affairs, Government of India, "Swachh Bharat Mission–Urban 2.0: Making Cities Garbage Free," New Delhi, India, 2021. [Online]. Available: https://ulb.arunachal.gov.in/pdf/schemes/SBM_Intro2.0.pdf (accessed Oct. 8, 2026).
- [10] Khan, S. Mishra, R. Das, and P. Singh, "Organic Waste Circular Economy for Viksit Bharat: Jobs, Investment, and Emissions Pathways to 2047," Council on Energy, Environment and Water (CEEW), New Delhi, India, May 2026.
- [11] S. K. Srichandan, S. Jena, S. K. Majhi, and K. Mishra, "A joint energy and resource aware task mapping in fog assisted cloud ecosystems," *Discover Internet of Things*, vol. 6, Art. no. 77, Apr. 2026, doi: 10.1007/s43926-026-00340-8.
- [12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [13] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks*, 1995.
- [14] P. R. Agarwal, K. Alwani, and A. K. Chaturvedi, "QoS-Aware Internet of Environmental Things Algorithm for IoT-Fog-Enabled Smart Waste Monitoring System," *ITEGAM-JETIA*, vol. 12, no. 58, pp. 999–1007, 2026.
- [15] M. Faccio, A. Persona, and G. Zanin, "Waste collection multi-objective model with real-time traceability data," *Waste Management*, vol. 31, no. 12, pp. 2391–2405, 2011.
- [16] S. Longhi, D. Marzioni, E. Alidori, G. Buo, M. Prist, M. Grisostomi, and M. Pirro, "Solid waste management architecture using wireless sensor networks," *Sensors*, vol. 12, no. 5, pp. 5952–5977, 2012.
- [17] M. Arebey, M. A. Hannan, H. Basri, H. Abdullah, and R. A. Begum, "Integrated technologies for solid waste bin monitoring system," *Environmental Monitoring and Assessment*, vol. 177, pp. 399–408, 2011.
- [18] S. Misra, S. Roy, and M. S. Obaidat, "Smart waste management using IoT," *IEEE Systems Journal*, vol. 13, no. 4, pp. 1–10, 2019.
- [19] C. Stergiou, K. E. Psannis, B. B. Gupta, and Y. Ishibashi, "Security, privacy and efficiency of sustainable Cloud Computing for Big Data and IoT," *Sustainable Computing*, vol. 19, pp. 174–184, 2018.

- [20] C.-H. Hong and B. Varghese, "Resource Management in Fog/Edge Computing: A Survey," arXiv:1810.00305, 2018.
- [21] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog Computing for Sustainable Smart Cities: A Survey," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–43, 2017.
- [22] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.
- [23] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions," *ACM Computing Surveys*, vol. 54, no. 11, 2022.
- [24] S. Bansal, H. Aggarwal, and M. Aggarwal, "A Systematic Review of Task Scheduling Approaches in Fog Computing," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 9, 2022.
- [25] S. M. Hashemi, A. Sahafi, A. M. Rahmani, and M. Bohlouli, "Energy-Aware Resource Management in Fog Computing for IoT Applications: A Review, Taxonomy, and Future Directions," *Software: Practice and Experience*, vol. 54, no. 2, pp. 109–148, 2024.
- [26] H. Salem and G. H. Algaphari, "Resource Allocation in Fog Computing: A Systematic Review," *Journal of Science and Technology*, vol. 27, no. 2, pp. 9–31, 2023.
- [27] F. A. Saif, R. Latip, Z. M. Hanapi, M. A. Alrshah, and K. Shafinah, "Workload Allocation Towards Energy Consumption-Delay Trade-Off in Cloud-Fog Computing Using Multi-Objective NPSO Algorithm," *IEEE Access*, vol. 11, pp. 45393–45404, 2023.
- [28] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, "Multi-Objective Grey Wolf Optimizer Algorithm for Task Scheduling in Cloud-Fog Computing," *IEEE Access*, vol. 11, pp. 20635–20646, 2023.
- [29] 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, art. no. 103333, 2022.
- [30] Mahapatra, S. K. Majhi, K. Mishra, R. Pradhan, D. C. Rao, and S. K. Panda, "An Energy-Aware Task Offloading and Load Balancing for Latency-Sensitive IoT Applications in the Fog-Cloud Continuum," *IEEE Access*, vol. 12, pp. 14334–14349, 2024.
- [31] M. S. Kumar and G. R. Karri, "EEOA: Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework," *Sensors*, vol. 23, no. 5, art. no. 2445, 2023.
- [32] Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Multi-Objective Task Scheduling in Cloud-Fog Computing Using Goal Programming Approach," *Cluster Computing*, vol. 25, no. 1, pp. 141–165, 2022.
- [33] Y. Gao, X. Zhang, and H. Wang, "Predictive Offloading and Resource Allocation for Delay-Sensitive IoT Applications in Fog Computing," *IEEE Access*, vol. 8, pp. 185610–185623, 2020.
- [34] M. Alizadeh, M. A. Kaafar, and R. Buyya, "Task Scheduling Approaches in Fog Computing: A Systematic Review," *Journal of Systems Architecture*, vol. 107, Art. no. 101707, 2020.
- [35] P. Singh, A. Nayyar, and R. Kumar, "Fog Computing: A Taxonomy, Systematic Review, Current Trends and Research Challenges," *Journal of Network and Computer Applications*, vol. 178, Art. no. 102972, 2021.
- [36] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions," *ACM Computing Surveys*, vol. 54, no. 11, pp. 1–42, 2022.
- [37] H. Salem and G. H. Algaphari, "Resource Allocation in Fog Computing: A Systematic Review," *Journal of Science and Technology*, vol. 27, no. 2, pp. 9–31, 2023.
- [38] S. M. Hashemi, A. Sahafi, A. M. Rahmani, and M. Bohlouli, "Machine Learning-Based Resource Management in Fog Computing: A Systematic Literature Review," *Software: Practice and Experience*, vol. 54, no. 2, pp. 109–148, 2024.
- [39] M. Hemmati, A. Boukerche, and R. Buyya, "Toward a Unified Computing Paradigm: A Survey and Roadmap for Data Management in Integrated IoT, Fog, and Cloud Services," *Peer-to-Peer Networking and Applications*, vol. 18, no. 2, pp. 1–34, 2025.
- [40] M. Fahad, A. Al-Dubai, S. Harous, and A. Y. Zomaya, "Mobility-Aware Dynamic Task Offloading in Fog Computing: A Multi-Objective Discrete PSO Approach," *Concurrency and Computation: Practice and Experience*, vol. 38, no. 4, Art. no. e70665, 2026.