

# Energy and Cost-Aware Real-Time Task Scheduling with Deadline-Constraints in Fog Computing Environments

Mayssa Trabelsi<sup>a</sup> and Samir Ben Ahmed<sup>b</sup>

LIPSIC Laboratory, Faculty of Sciences of Tunis, University of Tunis ElManar, Tunis, Tunisia

**Keywords:** Internet of Things (IoT), Fog Computing, Task Scheduling, Real-Time, QoS Optimization.


**Abstract:** With the increasing demand for real-time processing for IoT applications, Fog computing becomes a crucial approach to overcome the limitations of centralized Cloud Computing. Given its decentralized structure, Fog computing enables faster response time, real-time processing, and reduced latency, making it particularly suitable for time-sensitive IoT applications. In this paper, we propose a novel approach called the "Energy-cost-aware task scheduling with a Deadline-constrained" (ECaTSD) algorithm for real-time task scheduling in a fog infrastructure. The main objective of the proposed algorithm is to minimize energy consumption and monetary costs under deadline constraints. The ECaTSD algorithm dynamically allocates incoming tasks to the most suitable fog nodes in real-time. It selects the fog node that meets deadline requirements with the least energy consumption and monetary cost in the infrastructure. Moreover, the proposed algorithm has been simulated using the iFogSim simulator. The algorithm's performance is evaluated using various criteria, such as the percentage of IoT tasks successfully meeting deadlines, energy consumption, monetary cost, and response time compared to other scheduling policies. ECaTSD algorithm shows high efficiency in meeting deadlines (99.58% completion rate) while being energy and cost-efficient.


## 1 INTRODUCTION

The evolution of smart devices, incorporating communication technologies and advanced processing capabilities, has given rise to an innovative paradigm known as the Internet of Things (IoT) (Dabiri et al., 2022). Projections indicate that the number of IoT devices worldwide may surpass 29 billion by the year 2030 (Vailshery, 2024). As a result, generating a large volume of real-time data often requires intensive computational processing within strict time constraints (Stavrinos and Karatza, 2019) (e.g., Healthcare and traffic monitoring IoT devices and sensors) (Stavrinos and Karatza, 2019) (Dabiri et al., 2022). However, due to the computational capability limitations of IoT devices, they offload their task requests to external devices, such as servers (Dabiri et al., 2022). IoT devices require real-time responses and low latency, making cloud computing less suitable for time-sensitive IoT tasks. This is primarily due to the high communication delay between IoT devices and the cloud servers (Azizi et al., 2022).

Fog Computing (FC) has emerged as a new paradigm, extending the Cloud Computing paradigm to the edge of the network, near where real-time IoT data is generated, (Peter, 2015). FC is a distributed computing paradigm that serves as an intermediary layer between the cloud and IoT devices. It provides cloud servers' computational and storage services closer to the network's edge (Dabiri et al., 2022).

The FC paradigm has heterogeneous computational resources that can be virtualized, similar to cloud computing (Stavrinos and Karatza, 2019). Consequently, a Fog Node (FN) can be a Virtual Machine (VM) (Group et al., 2016). Also, FC involves various devices such as embedded servers, cellular base stations, switches, access points, routers, gateways, and surveillance cameras. However, the computational capacity of resources, including processing power (CPU), storage, and memory in a fog layer, is generally limited compared to that in the cloud (Stavrinos and Karatza, 2019). Therefore, task scheduling becomes a challenge when choosing suitable resources while maintaining Quality of Service (QoS), considering the diverse characteristics of each task and the heterogeneity of resources essential

<sup>a</sup>  <https://orcid.org/0000-0002-1723-9486>

<sup>b</sup>  <https://orcid.org/0000-0002-4642-2108>

for processing time-sensitive tasks in a fog computing environment (Jamil et al., 2022).

When developing algorithms to address the task scheduling problem, it is crucial to consider QoS metrics such as latency, response time, energy consumption, and cost to optimize performance and satisfy the timing constraints of IoT applications. The task scheduling problem comes under NP-Hard problems, which become more complex with increased sensors and FNs. (Jamil et al., 2022).

In recent years, many research efforts have proposed scheduling algorithms for optimizing various performance metrics to achieve high QoS (Sharma et al., 2023)(Jayasena and Thisarasinghe, 2019)(Xu et al., 2020)(Dabiri et al., 2022)(Sultan Hajam, 2024)(Hoseiny et al., 2020). The authors in (Sharma et al., 2023) proposed an approach to address task scheduling, considering various metrics such as energy consumption, latency time, cost, and network usage. An algorithm based on ant colony optimization is proposed in (Xu et al., 2020) to solve the task scheduling problem to reduce energy consumption while satisfying the task deadline. The authors in (Jayasena and Thisarasinghe, 2019) proposed a task scheduling method in a fog computing environment considering energy consumption and cost of execution. A system model was proposed in (Dabiri et al., 2022) to address the job scheduling problem, aiming to jointly minimize the total deadline violation time of IoT jobs and the system's energy consumption. The authors in (Sultan Hajam, 2024) proposed an algorithm to minimize the monetary cost while considering the deadline requirements and priority of tasks. A real-time task scheduling algorithm was proposed in (Hoseiny et al., 2020) to minimize IoT users' computation costs and reduce provider violation costs.

This paper presents an online scheduling algorithm called the "Energy-cost-aware task scheduling with a Deadline-constrained" (ECaTSD) algorithm designed for scheduling real-time tasks in fog infrastructure (FI). The ECaTSD algorithm provides multi-objective optimization of task scheduling to optimize performance metrics, such as energy consumption, monetary cost, and the percentage of IoT tasks that successfully meet their deadline. It is implemented in the iFogSim simulator (Gupta et al., 2017) and compared with Random, Shortest Execution Time (SET), Power of Two Choices (Po2C), and Greedy for Energy (GfE) algorithms.

The major contribution of this paper can be summarized as follows.

- A scheduler for dynamically assigning tasks in each region of a hierarchical FI is developed.

- We formulate the task scheduling problem as a Mixed Integer Linear Programming (MILP) model to optimize the FNs' total energy consumption and cost while meeting the task deadlines.
- We have proposed a dynamic heuristic algorithm to solve the real-time task scheduling problem to optimize multiple objective metrics, reducing energy consumption and minimizing monetary cost under deadline constraints.
- We have evaluated the performance of the proposed algorithm with the iFogSim compared with Random, Shortest Execution Time (SET), Power of Two Choices (Po2C), and Greedy for Energy (GfE) algorithms.

The remainder of the paper is organized as follows. Section 2 presents the proposed architecture and problem formulation. Section 3 describes the proposed algorithm. Finally, we evaluate the proposed solution in Section 4 and conclude the paper in Section 5.

## 2 SYSTEM MODEL AND PROBLEM FORMULATION

This section outlines the proposed approach for real-time task scheduling in fog computing. The proposed architecture is presented followed by the formulation of the problem.

### 2.1 Proposed Architecture

Many studies highlight the importance of geographical location for optimizing the performance of IoT applications (van der Zee and Scholten, 2013) (Naas et al., 2017). Our architecture is inspired by the work of (Naas et al., 2017) in which a strategy was proposed to place IoT data on a FI to reduce the overall latency in large-scale applications. Indeed, the authors introduced a heuristic approach, iFogStorZ, to minimize the problem-solving time by relying on geographical location as a partitioning criterion. In this work, we adopt their solution due to its significant reduction of the problem-solving time. They opted to define Regional Point of Presence (RPOP) (FNs located in layer three (FL2) that are illustrated in our proposed approach (see Fig. 1)) as points of partitioning. Hence, the maximum number of partitions corresponds to the number of FNs located in layer three. However, FNs located in layer three can also be grouped to form a specific zone. In this paper, we choose to place a scheduler for each zone.

The architecture of a fog computing infrastructure consists of three layers: IoT devices layer, fog com-

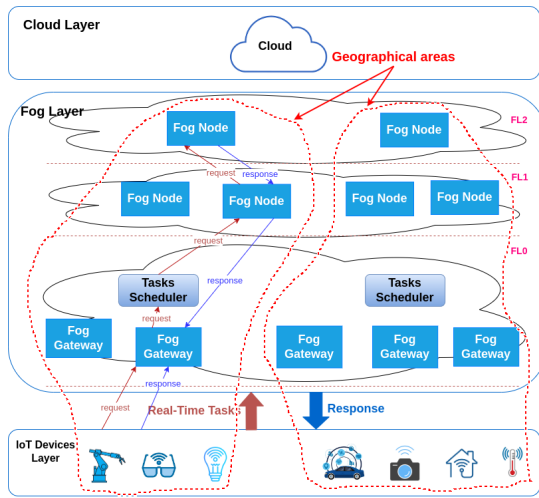


Figure 1: Proposed system architecture.

puting layer, and cloud computing layer, as shown in Fig.1 (Azizi et al., 2022)(Jamil et al., 2020)(Khan et al., 2022) .

- **First layer (IoT devices):** The first layer includes a large number of IoT devices, such as sensors and actuators. Sensors collect lots of time-sensitive data from the real world, demanding real-time processing, and offload them through gateways to appropriate fog/cloud nodes. The outputs of FNs are sent back as results to actuators, who take necessary actions within a specified deadline.
- **Fog Layer:** The fog layer is an intermediary tier between IoT devices and the cloud layer. It includes fog gateways, a scheduler, and FNs. The FNs are responsible for executing IoT tasks. However, they have limited computing, communication, and storage capabilities.
- **Cloud Layer:** The cloud is the top layer of the architecture. It is composed of powerful VMs with high computing power and storage capacity.

## 2.2 Problem Formulation

In this subsection, we formulate the multi-objectives problem of assigning real-time tasks in a hierarchical FI. First, the basic elements and decision variables of the optimization problem are introduced. Then, the response time, energy consumption, and cost formulas are described.

### 2.2.1 Basic Elements

Let  $n$  independent real-time tasks, denoted as  $T = \{T_1, T_2, \dots, T_n\}$ , submitted by IoT devices to the fog/cloud layer. Each task  $T_i \in T$  is characterized by

a set of attributes  $T_i = \{T_i^{length}, T_i^{deadline}, T_i^{in}, T_i^{out}\}$ , where  $T_i^{length}$  is the CPU length of task in  $MI$  (Millions of Instructions),  $T_i^{deadline}$  is the task's deadline requirement (in  $ms$  - milliseconds),  $T_i^{in}$  is the input file size (in  $KB$  - Kilobytes),  $T_i^{out}$  is the output file size (in  $KB$ ), and  $N_i$  is the network size of the task.

Assume a fog network, denoted as  $G = (F, L)$ , which includes several interconnected FNs. In this context,  $F = \{F_1, F_2, \dots, F_m\}$  represents a set of  $m$  FNs, and  $L = e_{uv} | u, v \in F$  denotes the set of communication links between nodes in the FI. Each node, represented as  $F_j$ , has some properties such as the CPU processing rate ( $F_j^{cpu}$ ) measured in MIPS (MI Per second) and the cost of processing per unit of time ( $C_{PT}$ ). Every link, denoted as  $e_{uv} \in L$ , is associated with characteristics including the propagation delay ( $e_{u,v}^P$ ) measured in  $ms$ , the network bandwidth ( $e_{u,v}^B$ ) is measured in  $Mbps$  (Megabytes per second), while its cost ( $C_{Nb}$ ) is measured in (\$) (Dollar).

### 2.2.2 Decision Variables

Let  $X_{m \times n}$  be the task assignment matrix, where the binary decision variable  $x_{i,j}$  means whether the task  $T_i$  is assigned to a FN  $F_j$ .

$$x_{i,j} = \begin{cases} 1, & \text{if } T_i \text{ is assigned to } F_j \\ & (\forall T_i \in T, \forall F_j \in F) \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

For each task  $T_i$ , we use a binary variable  $y_{uv}^i \in \{0, 1\}$  to indicate whether link  $e_{uv} \in L$  is selected for routing the task  $T_i$ . Thus, we have:

$$y_{u,v}^i = \begin{cases} 1, & \text{if link } e_{uv} \text{ is chosen for routing } T_i \\ & (\forall u, v \in F, \forall T_i \in T) \\ 0, & \text{Otherwise} \end{cases} \quad (2)$$

Finally, we define a binary variable  $z_i \in \{0, 1\}$  to indicate whether the deadline of task  $T_i$  is met. So, we have:

$$z_i = \begin{cases} 1, & \text{if } RT_{i,j} \leq T_i^{deadline} \\ 0, & \text{Otherwise} \end{cases} \quad (3)$$

### 2.2.3 Response Time

The response time  $RT_{i,j}$  for a given task  $T_i$  is the time needed to send the task request from an IoT device to the destination fog or cloud node and receive the results back. It includes the total transmission time  $T_T$ , execution time  $ET_{i,j}$ , and waiting time  $W_{i,j}$  in the queue of node  $F_j$ . The execution time  $ET_{i,j}$  of a task  $T_i$  on a FN  $F_j$  is calculated using Eq.4, where  $T_i^{length}$

represents the length of the  $i$ th task  $T_i$ , and  $F_j^{cpu}$  denotes the CPU processing rate of node  $F_j$ .

$$ET_{i,j} = \frac{T_i^{length}}{F_j^{cpu}}, \forall i \in T, \forall j \in F \quad (4)$$

To calculate the total transmission time  $TT_{i,j}^{Total}$  for task  $T_i$ , it is necessary to consider both the propagation delay  $PD_{i,j}$  and the transmission time  $TT_{i,j}$ . The transmission time  $TT_{i,j}$  is calculated using Eq.5, where  $T_i^{in}$  and  $T_i^{out}$  representing the sizes of the input and output files of task  $T_i$ , and  $e_{uv}^B$  denoting the bandwidth between two connected nodes  $F_u$  and  $F_v$ .

$$TT_{i,j} = \sum_{\forall e_{u,v} \in L} \frac{T_i^{in} + T_i^{out}}{e_{uv}^B} \times y_{u,v}^i, \forall i \in T \quad (5)$$

The propagation delay (Buyya et al., 2009) is the time required for a bit to travel from the source to the destination (see Eq.6).

$$PD_{i,j} = \sum_{\forall e_{u,v} \in L} (e_{uv}^P \times y_{u,v}^i), \forall i \in T \quad (6)$$

The total transmission time  $TT_{i,j}^{Total}$  can be defined as the sum of the propagation time and transmission time (see Eq.7).

$$TT_{i,j}^{Total} = 2 \times PD_{i,j} + TT_{i,j}, \forall i \in T \quad (7)$$

Waiting time refers to the duration between task submission and execution, including the time a task spends acquiring additional resources or waiting for specific events (Nikoui et al., 2020). The waiting time for a task  $T_k$  can be formulated using Eq.8, where  $T_k$  represents any task preceding the task  $T_i$ , whether it is waiting in the queue or currently running on the node.

$$W_{i,j} = \sum_{\forall k \in T} \sum_{\forall j \in F} (ET_{k,j} \times x_{k,j} \times x_{i,j}), \forall i \in T \quad (8)$$

Therefore, the response time for task  $T_i$  can be formulated (see Eq.9).

$$RT_{i,j} = TT_{i,j}^{Total} + ET_{i,j} + W_{i,j}, \forall i \in T \quad (9)$$

We use  $NT$  to calculate the number of tasks that meet their deadlines (see Eq.10).

$$NT = \sum_{\forall T_i \in T} z_i, \forall i \in T \quad (10)$$

The percentage of IoT tasks meeting the deadline requirement is defined as  $NT\%$  (see Eq.11).

$$NT\% = \frac{NT \times 100}{n} \quad (11)$$

## 2.2.4 Energy Consumption

The energy consumed by a Fog device is calculated in iFogSim, as (Jamil et al., 2020)(Khan et al., 2022)(Sharma et al., 2023) (see Eq.12).

$$EC_j = E_C + (T_C - T_L) \times HP, \forall j \in F \quad (12)$$

Where  $E_C$  is the current energy consumption,  $T_C$  is the current time,  $T_L$  is the update time of the last utilization, and  $HP$  is the host power in the last utilization.

The total energy consumption is calculated by the summation of the energy consumed by each FN (see Eq.13)

$$E^{total} = \sum_{\forall j \in F} EC_j \quad (13)$$

## 2.2.5 Cost

The cost represents the overall payment for these resource utilization (Nikoui et al., 2020). The costs for processing, denoted as  $Cost_{i,j}^E$ , and communication,  $Cost_{i,j}^C$ , are described as the following (Yadav et al., 2022) (Mokni et al., 2023) (Nikoui et al., 2020):

$$Cost_{i,j}^E = ET_{i,j} \times C_{PT}, \forall j \in F, \forall i \in T \quad (14)$$

$$Cost_{i,j}^C = TT_{i,j}^{Total} \times C_{Nb}, \forall j \in F, \forall i \in T \quad (15)$$

where the  $C_{PT}$  and  $C_{Nb}$  represents the cost of processing per unit of time price in destination node and cost of network bandwidth. Hence, the total cost  $Cost_{i,j}$  of a task  $T_i$  on a node  $F_j$  is calculated using Eq.16.

$$Cost_{i,j} = Cost_{i,j}^E + Cost_{i,j}^C, \forall j \in F, \forall i \in T \quad (16)$$

The total cost is calculated using Eq.17.

$$Cost^{Total} = \sum_{i=1}^n \sum_{j=1}^m Cost_{i,j} \times x_{i,j} \quad (17)$$

## 2.2.6 Problem Formulation

We aim to minimize energy consumption and monetary cost while meeting tasks' deadlines. Therefore, we formulate our problem mathematically as a multi-objective optimization problem.

$$Min : E^{Total} + Cost^{Total} \quad (18)$$

With the following constraints:

$$\sum_{\forall j \in F} x_{i,j} = 1, \forall i \in T \quad (19)$$

$$x_{i,j}, y_{u,v}^i, z_i \in \{0, 1\}, \forall i \in T, \forall j \in F \quad (20)$$

$$RT_{i,j} \leq T_i^{deadline}, \forall i \in T \quad (21)$$

Constraint 19 ensures that each task is assigned to only one FN. Constraint 20 defines the domain of the decision variables. Finally, constraint 21 ensures that the deadline for each task must be met.

The task scheduler selects the appropriate computing node by assigning each node a QoS score. This selection process ensures the optimization of QoS parameters: energy consumption and monetary cost. The QoS score is formulated using Eq.22.

$$Score_{i,j} = w_E \times EC_j + w_C \times Cost_{i,j} \quad (22)$$

Where  $Score_{i,j}$  represents the QoS score of each task assigned to a computing node, a lower  $Score_{i,j}$  value indicates that the FN  $F_j$  is more suitable for executing task  $T_i$ . The sum of the weight factors must be equal to 1, as expressed in Eq.23.

$$w_E + w_C = 1 \quad (23)$$

Where  $w_E = 0.5$  and  $w_C = 0.5$  are weight factors representing the substantial impact of the QoS parameters: energy consumption and cost in a specified node, respectively.

### 3 PROPOSED ALGORITHM

In this section, we propose ECaTSD, an energy-cost-aware task scheduling with a deadline-constrained, algorithm for solving real-time task scheduling problems in a fog computing infrastructure.

ECaTSD is a dynamic heuristic scheduling algorithm that depends on the states of system resources and the arrival of real-time tasks. The algorithm is in online mode, which processes unpredictable, independent tasks as they arrive in real-time without having complete knowledge of future inputs. It adapts and makes decisions based on the information available from the current system status at each step.

The ECaTSD aims to choose FNs with response times below the task's deadline. Within this subset of nodes satisfying the time constraint, the algorithm assigns the task to the node with the minimum QoS score, including energy consumption and cost. The mathematical model uses the QoS score to choose the optimal node, as per Eq. (22).

The pseudocode of the proposed algorithm is presented in Algorithm 1. Algorithm 1 introduces the proposed ECaTSD algorithm designed to address the task scheduling problem in a fog environment. The algorithm takes two inputs: a list of the currently incoming real-time tasks  $List_{T_i}$  and the list of the computing nodes  $List_F$  in its zone.

First, three empty lists are generated:  $L1$ ,  $L2$ , and  $L3$  (line 1).

Algorithm 1: ECaTSD algorithm.

---

**Data:**  $List_{T_i}$ ,  $List_F$   
**Result:**  $T_i \rightarrow F_j$

```

1 L1:=  $\emptyset$ , L2:=  $\emptyset$ , L3:=  $\emptyset$ ;
2 sort  $List_{T_i}$  in ascending order of deadline  $T_i$ ;
3 foreach  $T_i \in List_{T_i}$  do
4   L1:=  $\emptyset$ ;
5   foreach  $F_j \in List_F$  do
6     calculate  $RT_{i,j}$  using eq.(9);
7     L1 := L1  $\cup$   $RT_{i,j}$ ;
8   end
9   L2:=  $\emptyset$ ;
10  foreach  $RT_{i,j} \in L1$  do
11    if  $RT_{i,j} \leq T_i^{deadline}$  then
12      L2 := L2  $\cup$   $F_j$ ;
13    end
14  end
15  L3:=  $\emptyset$ ;
16  if L2  $\neq \emptyset$  then
17    foreach  $F_j \in L2$  do
18      calculate  $EC_j$  using eq.(12);
19      calculate  $Cost_j$  using eq.(16);
20      calculate  $Score_{i,j}$  using eq.(22);
21      L3 := L3  $\cup$   $Score_{i,j}$ ;
22    end
23    allocate  $T_i$  on  $F_j \in L3$  with the
      minimum  $Score_{i,j}$ ;
24  else
25    allocate  $T_i$  on  $F_j \in L1$  with the
      minimum  $RT_{i,j}$ ;
26  end
27 end

```

---

Next, the set of tasks in List  $List_{T_i}$  is sorted in ascending order based on their predefined deadlines (line 2). For each task, the algorithm attempts to find a suitable FN for the selected task  $T_i$  within the main loop (lines 3-27). Firstly, the list  $L1$  is reinitialized inside the loop (line 4). Subsequently, the second loop (lines 5-8) calculates the response time of each task  $T_i$  on each node  $F_j$  and adds them to the List  $L1$ . Afterwards, the list  $L2$  is reinitialized (line 9), and the algorithm iterates over the List  $L1$  (line 10), searching among all the FNs to check if a FN  $F_j \in List_F$  can meet the deadline of a task  $T_i$ . If the response time  $RT_{i,j}$  on node  $F_j$  is less than the deadline of  $T_i$ , the algorithm adds it to the deadline-satisfied List, i.e.,  $L2$  (lines 11-13).

The algorithm reinitializes the List  $L3$  (line 15) and checks if  $L2$  is not empty, meaning at least one FN can meet the task's deadline requirement  $T_i$ . At this point, the algorithm calculates the energy consumption using Eq.12, cost using Eq.16, and QoS score us-

ing Eq.22 of each FN, which is then added to the list  $L3$  (lines 17 to 22). Finally, the task  $T_i$  will be allocated to a FN with the least score  $Score_{i,j}$  in List  $L3$  (line 23). If the List  $L2$  is empty (no node satisfies the task's deadline), the task will be sent to the FN that has the least response time  $RT_{i,j}$  (line 25).

## 4 EXPERIMENTS AND RESULTS ANALYSIS

In this section, we discuss the performance of the proposed algorithm. The results are compared with other strategies, and the simulation settings are presented in subsection 4.1.

To evaluate the performance of the proposed algorithm, we compare it with other baseline algorithms : (1) Random, (2) Shortest Execution Time (SET) (Dutton et al., 2008), (3) Power of Two Choices (Po2C) (Hoseiny et al., 2020), and Greedy for Energy (GfE) (Xu et al., 2020).

### 4.1 Simulation Setting

The proposed algorithm was implemented in the iFogSim simulation tool (Gupta et al., 2017), using the Eclipse IDE environment (version 2023-09). Simulation experiments were conducted on a PC Intel with Core i7-2670 CPU 2.20 GHz, 3.7 GB RAM, and Ubuntu 20.04.5 LTS.

The simulated infrastructure is modeled based on the architecture shown in Fig.1. We consider eight IoT devices. Each device has a sensor with a distinct periodic frequency  $Freq_i$ , a deadline and one actuator. The infrastructure consists of eight fog gateways in Fog Level 0 (FL0), as each sensor is connected to a fog gateway, four FNs in Fog Level 1 (FL1), two in Fog Level 2 (FL2), and finally, one cloud server in the cloud layer, equipped with processing power ranging between 1000 and 40000 MIPS. Table 1 provides the parameter settings of the fog and cloud nodes. Table 2 presents the parameter settings for IoT tasks.

### 4.2 Results and Discussion

Here, we present the simulation experiments' results with the various settings outlined in subsection 4.1.

In real-time task scheduling in a FI, the proposed algorithm's performance is evaluated based on the percentage of IoT tasks that successfully meet their deadlines  $NT\%$ .

Further, the evaluation considers the average energy consumption (in  $MJ$ ) on FNs, the average monetary cost (\$) on FNs, the average response time (ms)

Table 1: Parameter settings for computing nodes.

Parameter	Value
Rate per MIPS at FN	0.00
Rate per MIPS at Cloud	0.01
Latency from fog to cloud layers (ms)	1200
Latency in fog layer (ms)	[50-500]
Latency from device to fog layers (ms)	10
$C_{PT}$ on cloud per time unit (\$/s)	[1- 2]
$C_{PT}$ on fog per time unit (\$/s)	[0.2-0.5]
$C_{NB}$ at fog per data unit (\$/s)	[0.01-0.02]
$C_{NB}$ at cloud per data unit (\$/s)	[0.05-0.01]
Propagation delay between FNs (ms)	[1 - 3]
Bandwidth of communication links (Mbps)	[100-1000]

Table 2: Characteristics of IoT tasks.

Parameter	Range of values
Number of instructions (MI)	[80000 - 300000]
Deadline (ms)	[2000 - 2500]
Frequency (ms)	[50 - 280]
Input file size (kb)	[50000 - 150000]
Output file size (kb)	3

of IoT devices, and the system's execution time (in ms).

$x_{i,j}$  in LF1 indicates the percentage of tasks assigned to LF1, while  $x_{i,j}$  in LF2 indicates the percentage of tasks assigned to LF2. Table 3 lists the performance results of the ECaTSD and other algorithms concerning deadline constraints and QoS metrics.

The results show that our algorithm ECaTSD effectively performs real-time task scheduling, meeting deadlines for 100% of the tasks. This achievement surpasses the performance of other algorithms. The GfE strategy attains a success rate of 78%, the SET strategy reaches 70%, and the Po2C strategy achieves 47.13%. However, the Random algorithm obtains the lowest percentage at 41.64%.

The Random strategy exhibits higher energy consumption ( $0.662 \times 10^7$  (MJ)) on FNs compared to Po2C ( $0.608 \times 10^7$  (MJ)), SET ( $0.612 \times 10^7$  (MJ)), and ECaTSD ( $0.623 \times 10^7$  (MJ)) strategies. In contrast, the GfE strategy demonstrates the lowest energy consumption on FNs ( $0.599 \times 10^7$  (MJ)). The increased number of assigned tasks  $x_{i,j}$  to LF2 increases the monetary cost. The load balancing between FNs in LF1 and LF2 may explain the minimal average response time observed for the ECaTSD algorithm. The monetary cost includes not only communication between the source and destination but also the processing of tasks on node  $F_j$ , taking into account the length of the task.

The proposed algorithm selects FNs that follow the deadline constraints. As a result, the cost and energy results are dependent on these nodes. Fig.2 demonstrates the percentage of tasks that have been

Table 3: QoS parameters' results of ECaTSD compared to other scheduling algorithms.

Metrics	Algorithms				
	Random	SET	Po2C	GfE	ECaTSD
$x_{i,j}$ in LF1 (%)	67.33	0	77.8	0	33.91
$x_{i,j}$ in LF2 (%)	32.66	100	15.12	100	66.08
$NT^{\%}$ (%)	41.64	70	47.13	78	100
$E^{FNs}$ (MJ)	$0.662 * 10^7$	$0.612 * 10^7$	$0.608 * 10^7$	$0.599 * 10^7$	$0.623 * 10^7$
$E^{Cloud}$ (MJ)	$1.4 * 10^7$	$1.34 * 10^7$	$1.342 * 10^7$	$1.335 * 10^7$	$1.386 * 10^7$
$Cost^{FNs}$ (\$)	7.919	9.6923	7.259	9.0780	6.174
Average Response Time (ms)	1949.625	4570.25	5005.25	4970.25	1805.5
Execution Time (ms)	362	440	368	413	354

assigned successfully and have met their deadlines. This varies with the number of tasks, and is applicable for all strategies. The SET scheduling algorithm achieves a 100% task satisfaction rate for up to 200 tasks. Meanwhile, the proposed task assignment approach achieves a 100% satisfaction rate for up to 400 assigned tasks. Only two tasks fail even when 480 tasks are assigned, resulting in a 99.58% task satisfaction rate.

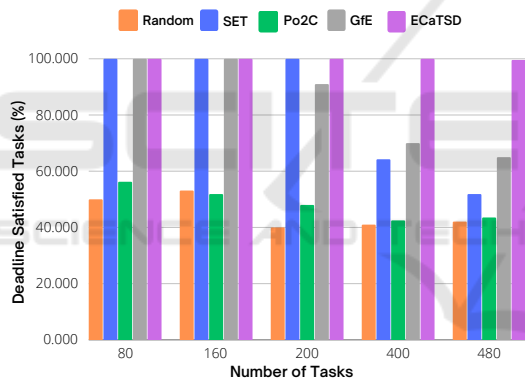


Figure 2: Percentage of tasks that meet their deadline by varying task numbers.

Fig.3 shows the average energy consumption of FNs. The x-axis represents the number of tasks, while the y-axis shows the total energy consumption in MJ. As shown, the Random strategy has a higher average energy consumption compared to other algorithms.

The illustration in Fig.4 shows the variation of total cost on FNs as the number of tasks changes. Among the different algorithms, SET appears to be the most cost-effective as it assigns most tasks to FL1. On the other hand, the ECaTSD algorithm aims to balance tasks across layers to meet time constraints, which leads to an average cost compared to other algorithms. This approach ultimately increases the costs.

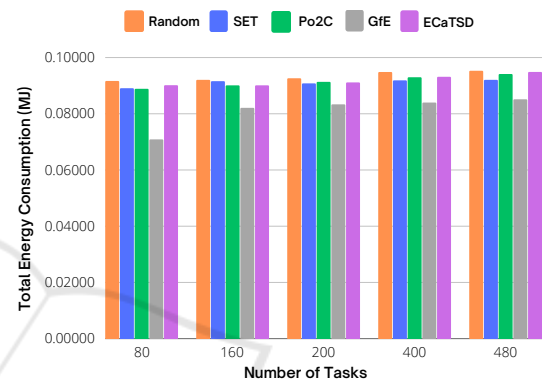


Figure 3: Total energy consumption by varying task numbers.

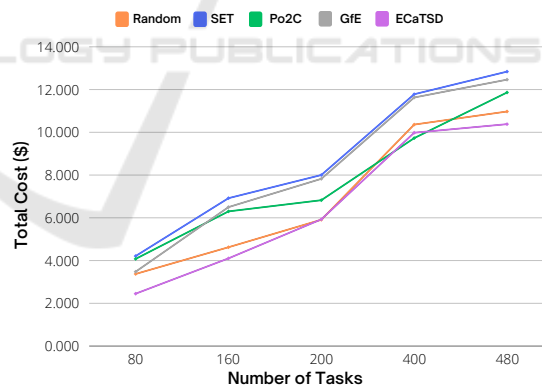


Figure 4: Total cost by varying task numbers.

## 5 CONCLUSION

This paper proposed a Energy-cost-aware task scheduling with a Deadline-constrained (ECaTSD) algorithm in Fog Infrastructure. We formulated the task scheduling problem as multi-objective problem and solved it using a dynamic heuristic algorithm to ensure high QoS. ECaTSD aims to allocate the tasks dynamically and in real-time, on the most suitable

fog nodes. The algorithm selects all the fog nodes that satisfy the task's requirements and then chooses the node that jointly minimizes the energy consumption and the cost. The proposed algorithm performance is evaluated according to the QoS criteria and compared to other online scheduling policies such as Random, Shortest Execution Time (SET), Power of Two Choices (Po2C), and Greedy for Energy (GfE) algorithms. The results show that ECaTSD has very encouraging results regarding the percentage of real-time tasks completed within their deadline, compared with the other algorithms. In future works, a deep reinforcement learning approach could be adopted for real-time scheduling.

## REFERENCES

- Azizi, S., Shojafar, M., Abawajy, J., and Buyya, R. (2022). Deadline-aware and energy-efficient iot task scheduling in fog computing systems: A semi-greedy approach. *J Netw Comput Appl.*, 201:103333.
- Buyya, R., Ranjan, R., and Calheiros, R. N. (2009). Modeling and simulation of scalable cloud computing environments and the cloudsims toolkit: Challenges and opportunities. In *International conference on high performance computing & simulation*, pages 1–11. IEEE.
- Dabiri, S., Azizi, S., and Abdollahpouri, A. (2022). Optimizing deadline violation time and energy consumption of iot jobs in fog–cloud computing. *Neural Computing and Applications*, 34(23):21157–21173.
- Dutton, R. A., Mao, W., Chen, J., and Watson, W. (2008). Parallel job scheduling with overhead: A benchmark study. In *International conference on networking, architecture, and storage*, pages 326–333. IEEE.
- Group, O. C. A. W. et al. (2016). Openfog architecture overview. *White Paper OPFWP001*, 216:35.
- Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296.
- Hoseiny, F., Azizi, S., and Dabiri, S. (2020). Using the power of two choices for real-time task scheduling in fog-cloud computing. In *4th International Conference on Smart City, Internet of Things and Applications (SCIOT)*, pages 18–23. IEEE.
- Jamil, B., Ijaz, H., Shojafar, M., Munir, K., and Buyya, R. (2022). Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(11s):1–38.
- Jamil, B., Shojafar, M., Ahmed, I., Ullah, A., Munir, K., and Ijaz, H. (2020). A job scheduling algorithm for delay and performance optimization in fog computing. *Currency and Computation: Practice and Experience*, 32(7):e5581.
- Jayasena, K. and Thisarasinghe, B. (2019). Optimized task scheduling on fog computing environment using meta heuristic algorithms. In *IEEE International Conference on Smart Cloud*, pages 53–58. IEEE.
- Khan, A., Abbas, A., Khattak, H. A., Rehman, F., Din, I. U., and Ali, S. (2022). Effective task scheduling in critical fog applications. *Scientific Programming*, 2022:1–15.
- Mokni, M., Yassa, S., Hajlaoui, J. E., Omri, M. N., and Chelouah, R. (2023). Multi-objective fuzzy approach to scheduling and offloading workflow tasks in fog–cloud computing. *Simulation Modelling Practice and Theory*, 123:102687.
- Naas, M. I., Parvedy, P. R., Boukhobza, J., and Lemarchand, L. (2017). ifogstor: an iot data placement strategy for fog infrastructure. In *IEEE IC FEC*, pages 97–104. IEEE.
- Nikoui, T. S., Balador, A., Rahmani, A. M., and Bakhshi, Z. (2020). Cost-aware task scheduling in fog-cloud environment. In *CSI/CPSSI International Symposium on RTEST*, pages 1–8. IEEE.
- Peter, N. (2015). Fog computing and its real time applications. *Int. J. Emerg. Technol. Adv. Eng*, 5(6):266–269.
- Sharma, O., Rathee, G., Kerrache, C. A., and Herrera-Tapia, J. (2023). Two-stage optimal task scheduling for smart home environment using fog computing infrastructures. *Applied Sciences*, 13(5):2939.
- Stavriniades, G. L. and Karatza, H. D. (2019). A hybrid approach to scheduling real-time iot workflows in fog and cloud environments. *Multimedia Tools and Applications*, 78:24639–24655.
- Sultan Hajam, S. (2024). Deadline-cost aware task scheduling algorithm in fog computing networks. *International Journal of Communication Systems*, page e5695.
- Vailshery, L. (2023, Accessed: January 11, 2024.). Number of internet of things (iot) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030. <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>.
- van der Zee, E. and Scholten, H. (2013). Application of geographical concepts and spatial technology to the internet of things. *Research Memorandum*, 33.
- Xu, J., Sun, X., Zhang, R., Liang, H., and Duan, Q. (2020). Fog-cloud task scheduling of energy consumption optimisation with deadline consideration. *International Journal of Internet Manufacturing and Services*, 7(4):375–392.
- Yadav, A. M., Tripathi, K. N., and Sharma, S. (2022). A bi-objective task scheduling approach in fog computing using hybrid fireworks algorithm. *The Journal of Supercomputing*, 78(3):4236–4260.