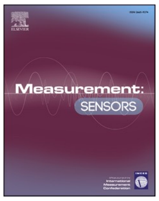
Contents lists available at ScienceDirect

Measurement: Sensors

journal homepage: www.sciencedirect.com/journal/measurement-sensors



An effective technique to schedule priority aware tasks to offload data on edge and cloud servers

Malvinder Singh Bali[a], Kamali Gupta[a], Deepali Gupta[a], Gautam Srivastava[b,c,d,*], Sapna Juneja[e], Ali Nauman[f]

[a] Chitkara University Institute of Engineering & Technology, Chitkara University, Punjab, India
[b] Department of Math and Computer Science, Brandon University, Canada
[c] Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan
[d] Department of Computer Science and Math, Lebanese American University, Beirut, 1102, Lebanon
[e] Department of Computer Science, KIET Group of Institutions, Delhi NCR, Ghaziabad, 201206, India
[f] Department of Information and Communication Engineering, Yeungnam University, Republic of Korea

ARTICLE INFO

Keywords:
Queue model
Data offloading
Edge computing
Scheduling

ABSTRACT

Recent advancements in the Internet of Things (IoT) have enhanced the quality of life globally. Billions of devices are brought under the ambit of IoT to make them smarter. IoT-based applications are generating voluminous data and managing this widespread amount of data in real-time through Cloud Technology, which offers high computational and storage facilities. However, sending all data to the cloud can bring serious concerns for applications, which are critical and require instant action without any delay. Edge computing has recently emerged as an effective technology to handle the instant processing of tasks of IoT-based applications locally. Additionally, an important concern in IoT networks is response to emergency tasks on time to increase the performance of large-scale IoT systems. As such, scheduling of tasks becomes vital, where emergency and non-emergency tasks can be prioritized to offload data to the nearby edge and cloud servers respectively and enhance Quality of Service (QoS). The execution order of tasks and allocating resources for computation to avoid delays are two of the most important factors that must be addressed during task scheduling in Edge Computing. With the aforementioned issues, we design a Priority aware Task Scheduling (PaTS) algorithm for sensor networks to schedule priority aware tasks to offload data on edge and cloud servers. The problem is formulated as a multi-objective function and the efficiency of the proposed algorithm is evaluated using the Bio-inspired NSGA-2 technique. The overall improvement for average queue delay, computation time, and energy obtained for 200 tasks is 17.2%, 7.08% and 11.4%, respectively. The results obtained show significant improvement when compared with the benchmark algorithms demonstrating the effectiveness of the proposed solution. Similarly, comparative results for tasks when increased from 200 to 1000 tasks also shows subsequent improvements.

1. Introduction

An emerging technology known as the Internet of Things (IoT) connects people with machines and objects. With IoT, machines can be made smarter. It is anticipated that it will continue to expand its growth to enhance the quality of life globally [1]. As per the Cisco report, by the year 2022, 14.3 billion gadgets are projected to be linked to the Internet [2]. Considering the data produced by billions of IoT devices, managing this massive, widespread amount of data in real-time has emerged as the mainchallenge. Cloud Computing has played a pivotal role in offering high computational and storage facilities, and on-demand use of a shared resource pool over the last decade [3]. Since Cloud Computing offers abundant computer resources, sending all data to the cloud may result in increased traffic, causing congestion and delays at unpredictable times. This may lead to latency issues and decrease the quality of users' experience [4].

Massive amounts of data are introduced with an increase in connected devices, which presents another problem for today's networks to successfully handle [5,6]. Edge Computing helps to remove the shortcomings of Cloud Computing. Intensive processing tasks in IoT are

* Corresponding author. Department of Math and Computer Science, Brandon University, Canada.
E-mail addresses: malvinder.singh@chitkara.edu.in (M.S. Bali), kamali.singla@chitkara.edu.in (K. Gupta), deepali.gupta@chitkara.edu.in (D. Gupta), srivastavag@brandonu.ca (G. Srivastava), sapnajuneja1983@gmail.com (S. Juneja), anauman@ynu.ac.kr (A. Nauman).

https://doi.org/10.1016/j.measen.2023.100670
Received 14 October 2022; Received in revised form 28 November 2022; Accepted 7 January 2023
Available online 10 January 2023

transferred to the local edge servers to remove latency issues [7]. Since it has received so much attention in recent years, governments, technology pioneers, and researchers are working to expand the use of edge computing [8]. In Edge Computing, storage facilities, resources on the edge network, and computational requirements are limited. So, scheduling tasks in Edge Computing has become vital to enhance the Quality of User Experience (QoE). The execution order of tasks on time and allocating resources for computing to avoid delays are the two most important factors that should be addressed during the scheduling of tasks on Edge-based computing networks.

1.1. Motivation

With the rapid expansion of the Industrial Internet of Things (IIoT), there is a huge growth in networks, and traditional cloud servers fail to meet the emergency real-time deadlines and reliability needs of Industrial IoT for transmission of data and processing. One such instance is a Smart factory, where remote equipment monitoring devices generate huge amounts of data and upload all data to the cloud for processing, which will consume a lot of bandwidth, operational cost will be high, there will be high load on the server and overall task processing time will increase, causing latency issues [8]. Some emergency tasks require instant action, such as the urgent shutdown of industrial boilers due to temperature rise. More negative implications will result from delaying the delivery of this data to the cloud for quick processing and sending back the appropriate response. Hence, the aforementioned issues need the adoption of an effective scheduling concept where high priority tasks will be scheduled first and offloaded to nearby Edge servers for instant action and low priority tasks can be scheduled after a certain threshold time and offloaded to Cloud servers as the latency factor of low priority tasks is negligible. The entire process will help to offload data from low-powered IoT devices to slightly higher resource-rich devices using an optimal scheduling policy. All of this involves offloading the task to nearby Edge servers for processing [9,10]. Edge computing, a research solution for the aforementioned issues, involves sending a portion of IIoT data to edge networks. Some IIoT applications are gradually implementing edge computing (e.g., distant equipment surveillance [11–13], preventing future problems [14], and quality control [15]).

2. Related work

In Edge Computing, to avoid latency issues. delay-based sensitive tasks are sent to the nearest edge servers. Edge servers are also resource-deficient in terms of computing power, storage and network capacity. So, efficient scheduling of tasks to maximize QoS becomes essential. Task scheduling in Edge Computing has been a challenge for several reasons. First, due to the dynamic condition of wireless channels between end devices and edge nodes. Also, different tasks from various IoT applications have different sizes, delays, and arrival rates, making it more challenging.

So, in task scheduling, two important problems need to be addressed. Delay-sensitive tasks need to be scheduled first based on priority factors and second, on the instant allocation of resources to critical/delay-sensitive tasks near the network's edge for execution, which means mapping tasks to nearby edge nodes should be done effectively. Lots of studies based on task scheduling have been attempted.

Some of the existing studies based on task scheduling focussed on optimizing energy usage. In Ref. [16], the authors suggest energy management and task planning solutions for energy delivery from the edge server or base station to energy-harvesting IoT devices. Scheduling of tasks and allocation of energy is carried out on IoT and Edge servers respectively, to gain energy efficiency. In Ref. [17], the authors presented coupled radio and processing resource scheduling techniques employing OFDMA for mobile edge computing systems, which are efficient and almost ideal for saving energy for mobile devices. In Ref. [18], the authors designed a novel Data offloading approach to minimize energy and latency issues. Simulation results exhibit improvements of up to 20–30% over existing algorithms. The authors in Ref. [19], focussed on scheduling computationally intensive tasks in edge-based IoT systems where the execution order of task and task allocation via Virtual Machines (VMs) are optimized together. The problem is formulated using a Markox Decision Process (MDP) and the problem is resolved using the Deep Reinforcement Learning (DRL). The results show that the algorithm performs better compared to benchmark algorithms. The authors in Ref. [20] proposed a general model to overcome the problem of arbitrary order of jobs and second offloading to unassociated servers with increased delay in uploading and downloading. The model's main goal was to reduce the overall response time for all operations. In Ref. [21], the authors proposed a resource scheduling algorithm to minimize the total communication and computation delays using cellular-based edge networks. The proposed a hybrid dynamic scheduling scheme (HDSS) in Ref. [22] incorporates dual scheduling algorithms (queue-based dynamic scheduling (QDS) and time-based dynamic scheduling (TDS)) to adjust changing dynamics of the system. The proposed system also designed a decision support function to select the best scheduling algorithm between QDS and TDS. Experts reveal that HDSS fits well for heterogeneous systems. In Ref. [23], using edge computing, a Cache-aware task scheduling approach is proposed. Finally, an integrated utility based cache placement technique is used. To increase the integrated utility value of caching, data chunks are cached on the best edge servers. Tasks are then planned following cache placement outcomes. In Ref. [24], researchers created a Mixed Integer Program (MIP) to formally define and formulate the Dynamic Task Offloading and Scheduling problem (DTOS) MIP. To address Task Scheduling in Fog Computing (TSFC) problem in IoT applications, in Ref. [25], a new energy-aware algorithm called the Marine Predators Algorithm (MPA) is developed. In addition to basic MPA, two variants of MPA have been suggested in this study to address the TSFC. In Ref. [26], the authors investigated edge computing online deadline-aware task dispatching and scheduling. To meet the increased deadlines, the authors suggested an online algorithm called DeDas that greedily schedules newly arriving jobs and analyses whether to replace certain current activities. In Ref. [27], the authors created a tiered structure where all the incoming tasks are scheduled according to the priority set, using the frequency of arrival of the tasks and execution deadline to decrease the average queue time of jobs in industrial sensor networks.

In addition to this, some papers have focussed on minimizing the overall execution delay of tasks. In Ref. [28], the authors created a resource allocation technique for multi-user offloading systems to reduce the average observed latency of the worst-case client. To reduce application completion time, the study in Ref. [29] examines data-aware task allocation, which jointly schedules network flows and tasks. The issue takes into account both the location of data and the amount of network bandwidth used when transferring data to plan jobs. As soon as IoT devices are configured and local tasks are put into servers, iterative heuristic MEC resource allocation (IHRA) [30] provides a method for job scheduling designed to decrease the duration of the entire execution by taking both cloud and edge servers into account. For Internet of Battery Less Things (IoBT) networks in Ref. [31], the authors suggested a cooperative in-network processing paradigm to achieve the lowest latency while generating dynamic ambient energy. The suggested methods for cooperative in-network processing consistently achieve lower latency when compared to the existing schemes, regardless of the data processing ratios and input data quantities. In Ref. [32], a heuristic-based data offloading technique is proposed for IIoT-based sensors to reduce the data migration capacity and enhance bandwidth by offloading data to the nearby Edge servers. Real-time experiments show the effectiveness of the proposed algorithm. In Ref. [33], a systematic survey was presented on data offloading approaches in IoT at edge and Fog Nodes. A smart framework is proposed to tackle data offloading issues. In Ref. [34], the authors proposed a smart Hydroponic approach for saffron cultivation using IoT.

From Table 1, it is observed that the existing studies have mainly focussed on minimizing energy during the task scheduling process. Nonetheless, the above studies have not focussed on minimizing the queue delay and prioritizing emergency tasks. In the proposed work, the main focus will be to cover all aspects, mainly minimizing the average queue delay of the task [35]during the scheduling process to overcome the latency issue.

3. Objectives and contributions

Using the aforementioned challenges as inspiration, a Priority Aware Task Scheduling (PaTS) algorithm [36] is designed to offload industrial-based sensor data. The primary contributions of this paper are as follows:

- Formulate a new policy for the Four Queue model [37] where very urgent and urgent tasks will be enqueued in the first and second queue respectively for offloading to the nearby edge server, keeping the latency factor into consideration. Moderate and non-urgent tasks will be enqueued in the third and fourth queue respectively for directly offloading to the cloud as the latency factor is low in such tasks [38].
- Optimize the Average Queue Delay [39] and Energy Consumption of incoming emergency tasks [40] produced by Industrial sensors using a multi-objective NSGA-II based Genetic algorithm.

The remainder of the paper is structured as follows. The architecture of the edge-based network is shown in Section 4. In Section 5, the oroposed PaTS approach is displayed. Section 6 provides examples of the experimental study of the proposed technique. Section 7 presents a conclusion at the end.

4. Problem formulation and system modelling

4.1. Network model

In Fig. 1, the system consists of several edge devices (E) and Industrial Sensors (S) Distributed randomly represented as $E = \{E_1, E_2 \dots \dots E_n\}$ and $S = \{S_1, S_2 \dots . S_n\}$ respectively [41]. Through G, a group of gateway devices denoted as $G = \{G_1, G_2 \dots G_n\}$ will help to communicate between Sensors and Edge devices [42]. The notations are shown in Table 2. The Gateway devices communication module collects information from the sensing device and then sends it to computer devices via the network. A lightweight communication protocol called Message Queuing Telemetry Transport (MQTT) is used in Gateways to transmit sensed data through dispersed computing devices like Edge devices. Wireless networks and IoT devices with limited resources benefit the most from the MQTT protocol [43-44].

The sensing device S periodically generates separate tasks $O = \{O_1, O_2 \dots O_n\}$ for computation. The two separate tasks order are shown as i < j which denotes that Oj P(O$_i$) i.e., Oi is Oj Predecessor [45]. The representation of the incoming tasks is depicted into 4 tuples i.e., $O_i \langle O_i^{CPU}, O_i^f, O_i^d, O_i^z \rangle$, $1 \le i \le n$; where $O_i^{\ CPU}$ denotes the amount of CPU required, O_i^d denotes the deadline bound, O_i^f denotes the frequency of arrival and O_i^z denotes the amount of data for task O_i. Depending upon the Binary Offloading policy [33], upcoming tasks are processed locally or offloaded to the active computing devices set V. = {E ∪ S}. Let us consider that $\Gamma(O, V) = R^{O^*|S \cup E \cup C}$ be a task allocation matrix, where $\Gamma(Oi, Vj) = \{(0,1)|O_i \in O, Vj \in (S \cup E \cup C\}$. Local gateway devices make scheduling and remote offloading decisions. To schedule tasks and offload data in the proposed scheme, a local gateway implemented through Raspberry Pi has been considered. Multiple Raspberry Pis are thought of as local gateway devices for reducing failure tolerance [46]. However, only one acts as a centralized gateway. The nearby device could serve as a gateway to control the failure rate of emergency tasks, if there is an issue with one gateway device (in this case, the Raspberry Pi).

4.2. Model for local execution

A task O_i must meet the CPU restriction of a local sensing device S_j to be processed by that device which means $Oi^{\ CPU} \le S_j^{\ CPU}$: $Sj \ \varepsilon \ S$

To process a task O_i, we need to know the CPU frequency of the Local Sensing device S_j defined as $F_j^{\ CPU} = O_i^Z * O_i^{Pr}$ where O_i^{Pr} stands for the O_i^{th} task's processing density and $O_i^{\ Z}$ stands for data size. Therefore, the total time taken by the O_i^{th} task to process on S_j^{th} sensing device is calculated as:

$$T_{ij}^S = \sum_{i=1}^n \Gamma(Oi, Vj) F_j^{\ CPU} \Big/ CF_j F_j^{CPU} \tag{1}$$

Here, CF_j^{CPU} is the computational frequency of the jth sensing device. To reduce power consumption of IoT sensing devices, it is important to calculate the dissipation of power [**34**] of the local S_j^{th} sensing device represented as:

$$E_{ij}^S = \propto F_i^{CPU} \left(CF_j^{CPU} \right)_2 \tag{2}$$

where ∝ is a constant called Switching Capacitance.

4.3. Model for remote execution

Due to the low processing power of local sensor devices, we offload some tasks to a remote server [47]. To calculate efficiency, the transmission rate of data and the data uploading time of a remote server [48], we need to frame mathematical formulas to calculate the results.

The following is an expression for the rate of data transmission from the ith local sensing device to the v_j^{th} computer device located remotely:

$$D_{ij}^{UP} = \beta_{ij}^{UP} \log_2 \left(1 + \frac{P_i^{trans} G_i^{Power}}{\gamma_j^{\ 2}} \right) \tag{3}$$

where β_{ij}^{UP} represents bandwidth utilized to send data from the ith sensing device to the jth computing device, the ith sensing device's data transmission rate is P_i^{trans} , G_i^{Power} is the maximum power utilized by the jth computing device, $j \in (R \cup S)$ and $\gamma_j^{\ 2}$ denotes the combined noise of the jth device and the ith sensing device's data transmission rate, respectively. Therefore, the time to upload data from the ith local sensing device to the jth computing device is calculated as:

$$T_{ij}^U = \frac{\sum_{i=1}^n \Gamma(O_i, v_j) O_i^2}{D_{ij}^{UP}} \tag{4}$$

The amount of energy used to send a task to device V_j is listed as:

$$E_{ij}^{\ U} = T_{ij}^U P_i^{\ trans}$$

Table 1
Comparative analysis with the existing algorithms.

Reference	Binary Offloading	Task Priority	Energy Consumption	Queue Delay
[19]	✓	x	X	X
[20]	✓	x	X	✓
[21]	x	x	X	✓
[22]	x	x	X	✓
[23]	x	x	✓	x
[24]	✓	x	x	✓
[25]	x	x	✓	x
[26]	✓	x	x	✓
[27]	✓	✓	✓	x
Proposed Work	✓	✓	✓	✓

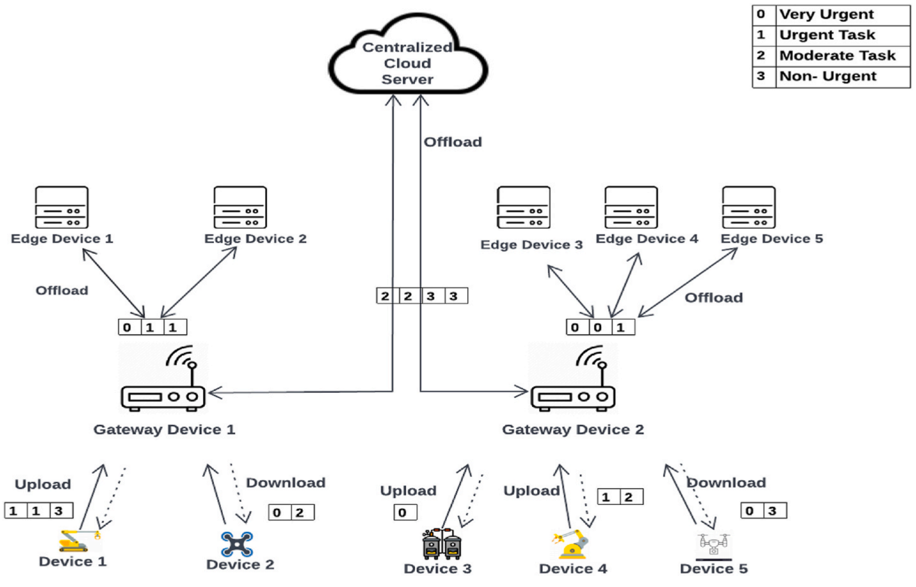


Fig. 1. Industrial edge network architecture.

Table 2
List of notations.

Symbol	Description
E	Edge Nodes
G	Gateway Device
C	Centralized Cloud server
O	Independent Task
V	Computing Device
S	Set of IoT Devices
O_i^{CPU}	CPU requirement
O_i^f	Arrival Frequency
O_i^d	Deadline Bound
O_i^z	Data Size

Through a local gateway device, every next task O_i gets transferred to an appropriate computing system with the frequency of CPU as CF_j^{CPU}. Consequently, the Processing Time on Device V_j is shown as

$$T_{ij}^{V} = \frac{\sum_{i=1}^{n} \Gamma(O_i, v_j) F_i^{CPU}}{CF_j^{CPU}} \tag{5}$$

On device V_j, the power dissipation E_{ij} is defined as follows:

$$E_{ij}^{V} = \propto F_i^{CPU} \left(CF_j^{CPU}\right)^2 \tag{6}$$

The selected V_j^{th} computing equipment begins transmitting the outcome back to the sensing device upon completion of the O_i^{th} task's processing. Similar to that, the sensing device's downloading transmission rate $D_{ji}^{\ down}$ from V_j is defined as:

$$D_{ji}^{down} = \beta_{ji}^{down} \operatorname{Log}_2\left(1 + \frac{P_i^{trans} G_j^{power}}{\gamma_i^2}\right) \tag{7}$$

A time estimate for data download via device j to device i is represented as:

$$T_{ji}^{D} = \frac{\sum_{1}^{N} \Gamma(O_i, v_j) O_i^2}{D_{ji}^{down}} \tag{8}$$

The amount of energy used to download O_i^{th} task from device V_j is indicated by:

$$E_{ji}^{\ D} = T_{ji}^{D} P_j^{trans} \tag{9}$$

4.4. *Queuing model*

Suppose we have a Scheduled System $T = \{T_1, T_2 \ldots T_n\}$ where Inter-arrival tasks between two consecutive tasks are distributed using the Poisson Distribution method $f(X) = P(X = x) = e^{-\lambda}\lambda^x/x!$ where e is the Eulers No, λ is the mean arrival rate and $\lambda > 0$.

The suggested sensor network design claims that each gateway device divides the task into four distinct groups into Very Urgent task (OVU), Urgent task (OU), Moderate task (OM) and Non-Urgent task (ONU) as shown in Fig. 1. Very Urgent is of high importance and seeks to support time-sensitive tasks with strict deadlines. To prevent lengthy offloading and downloading times when using a remote cloud, tasks are either handled locally on IoT devices or given to nearby edge devices. For these duties, a more dependable computing device is specifically assigned. An urgent task has an almost equivalent structure to a very urgent task, but the waiting time is slightly longer compared to a very urgent task. Moderate tasks have a soft deadline and a middle priority level. These tasks meet their deadline by negotiating total latency i.e., extra time may be given to finish the task processing. These tasks are processed on a cloud server. The non-urgent task has the lowest priority and is intended to enable time-consuming jobs with no deadline. Such jobs need a greater capacity of resources (such as memory and CPU frequency) for processing without a focus on decreasing latency. The centralized cloud server is typically used to offload tasks of this nature. The gateway devices construct non pre-emptive four level feedback queues with Very Urgent Queue (QVU), Urgent queue (QU), Moderate queue (QM) and Non-Urgent Queue (QNU)for further decision-making to reduce scheduling delays (i.e., not under starvation) [49].

The following is the expression for the Queuing delay T_{ij}^{Q} for a task O_i throughout its existence in a gateway device G_j.

$$\mathbf{T_{ij}^{Q}} = \begin{cases} W_i^{Q,VU} \text{if } O_i \in O^{V}U \\ W_i^{Q,VU} + W_i^{Q,U} \text{if } O_i \in O^{U} \\ W_i^{Q,VU} + W_i^{Q,U} + W_i^{Q,M} \text{if} O_i \in O^{M} \\ W_i^{Q,VU+} W_i^{Q,U} + W_i^{Q,M} + W_i^{Q,NU} \text{if} O_i \in O^{N}U \end{cases} \tag{10}$$

Where $W_i^{\ Q,\ VU}$, $W_i^{\ Q,\ U}$, $W_i^{\ Q,\ M}$ and $W_i^{\ Q,\ NU}$ represent the waiting time of

Q^{VU}, Q^{U}, Q^{M} and Q^{NU} Queue [**35–37**].

4.5. Problem formulation

The total energy used while offloading task O_i to the cloud server or Edge server includes Energy consumed while uploading, downloading and processing and is expressed as:

$$E_{ij}^{Total} = E_{ij}^{U} + E_{ij}^{D} + E_{ij}^{V} \tag{11}$$

When processing on a nearby sensing device or a distant server, a task O_i's overall energy consumption is expressed by the following formula:

$$E_{ij}^{Total} = \min\left(E_{ij}^{P}, E_{ij}^{Off}\right) \tag{12}$$

The proposed multi-objective with significant limitations is represented mathematically as follows:

$$\text{Minimize} \sum_{i=1}^{n} E_{ij}^{Total}(t) \tag{13}$$

$$\text{Minimize } T_{ij}^{U} + T_{ji}^{D} + T_{ij}^{Q}$$

Subjected to

$$E_{ij}^{U} \geq 0 \text{ and } E_{ji}^{D} \geq 0 \tag{13a}$$

$$T_{ij}^{U} + T_{ji}^{D} + T_{ij}^{Q} \leq T_{ij}^{\ max} \tag{13b}$$

$$\Gamma(O_i, v_j) \in \{0, 1\} \tag{13c}$$

$$\sum_{j \in |\nu|} \Gamma(O_i, v_j) = 1 \tag{13d}$$

$$\sum_{i \in |o|} \sum_{j \in |\nu|} \Gamma(O_i, v_j) \leq |\nu| \tag{13e}$$

where Equation (13a) indicates that while uploading and downloading data, the value cannot be zero. Equation (13b) states that the entire amount of time needed to complete a task (O_i) after it has been downloaded, uploaded, and queued should not exceed or be greater than the maximum delay (T_{ij} max). According to Equations (13c) and (13d), while a task O_i can make many requests for computation from different devices, every task must be given only to one computing device. The choice to offload the task is restricted by Equation (13e).

5. Energy efficient framework for data offloading (E^2FDO)

The E2FDO strategy is based on two modules, namely Priority aware Task Scheduling (PaTS) which uses a Multi-layered feedback queue technique to find a workable scheduling order and assign priorities to incoming tasks generated by industrial sensors, and the next Efficient Task Allocation Strategy will be used to map the scheduled task with suitable computing devices by using the effective and latest matching technique. This paper will completely discuss the policy to assign Priority to emergency and non-emergency tasks using the arrival frequency of tasks and their execution deadline.

5.1. Priority aware task scheduling (PaTS)

The primary function of this scheduling algorithm is to categorize the incoming tasks produced by sensor devices using the Utilization Factor (U_i) and allocate the task to their respective queues based on priority. A set of emergency information serves as the task identifier for each O_i in O like the frequency of arrival task (O_i^f) and execution deadline (O_i^e). Each incoming task priority is determined by a Utilization factor (U_i) and represented by $U_i = O_i^e / O_i^f$. Depending on this utilization factor, a task is differentiated as a Very Urgent task (O_i^{VUT}), Urgent task (O_i^{UT}), Moderate task (O_i^{MT}) and Non-Urgent Task (O_i^{NUT}). Tasks generated by sensor devices will be enqueued into the global queue in the First Come First Serve (FCFS) order of the Gateway (i.e., Raspberry Pi). A further priority of the task based on the utilization factor will be assigned to the task and later tasks will be enqueued into the respective queues as per their Priority Aware Scheduling Policy. To avoid starvation, if the waiting time of any queue exceeds its threshold, the task will be dequeued from that respective queue. To add novelty to the queue model, we have included two pairs of queues where urgent tasks will be handled by one pair of queues for further offloading to the Edge server and non-urgent tasks will be handled by another pair of queues for directly offloading to the cloud server. All the priority assignment and allocation of queues for the task is done on the gateway device as shown in Fig. 2.

5.2. A task priority is described in the following way

Definition - The task O_i coming from a sensor device is classified as a Very-Urgent task if $U_i < \frac{1}{4}$, or Urgent if $\frac{1}{4} < U_i < \frac{1}{3}$, or Moderate task if $\frac{1}{3} < U_i < \frac{1}{2}$ or a non-Urgent task if $U_i > \frac{1}{2}$.

Explanation- Consider a set of six tasks with the representation O= (O_1, O_2, O_3, O_4, O_5, O_6) where $O_1 = <4,4>$, $O_2 = <4,6>$, $O_3 = <3,8>$, $O_4 = <2,9>$, $O_5 = <1,2>$, and $O_6 = <4,5>$, respectively. As per the Utilization Factor criteria, Task O_i utilization is calculated as U = 1. Likewise, U = 0.6, U = 0.3, U = 0.2, U = 0.5, and U = 0.8, respectively. Because the Utilization level determines task priority, Task O_4 is categorized as a very urgent task. Under the definitiontask O_1, O_2 and O_6 are classified as non-Urgent task, O_3 as Urgent task, and O_5 as Moderate, respectively.

5.3. Task scheduling

The PaTS algorithm considers four queues to establish a Multilevel feedback Queue model with the most Urgent tasks sent to the Very High Priority queue, represented in Q_1. Similarly, urgent tasks are sent to the next queue, Q_2, which is slightly less priority as compared to Q_1. Moderate tasks were sent to the queue Q_3 with Intermediate priority and non-Urgent tasks were sent to the Low Priority Queue Q_4. Additionally, Dequeue () is utilized to determine a workable schedule order for receiving data according to priority, and the following queue time quantum is defined next.

5.3.1. At Schedular 1

Rule 1: If $Q_2^{\ UT} \neq \varphi$ and $W_i^{Q,\ UT} \geq \widetilde{Q}^{UT}$, $\forall_i \in O_i^{\ UT}$ then task O_i should therefore be given higher priority and pushed in $O_i^{\ VUT}$. Dequeue (O_i^{UT}) Q^{VUT} i.e remove the task from the Urgent queue and put it in the Very Urgent queue on an FCFS basis.

Rule 2: When $Q_1^{\ VUT} = \varphi$ and $Q^{UT} \neq \varphi$, dequeue task (O_i) from Urgent queue for directly offloading at edge server in FCFS basis.

Rule 3: When $Q_1^{\ VUT} \neq \varphi$ and $Q^{UT} = \varphi$, dequeue task (O_i) from Very Urgent queue O_i^{VUT} for offloading at edge server through a gateway in FCFS basis.

5.3.2. At Schedular 2

Rule 1: When $Q_4^{\ NUT} \neq \varphi$ and $W_i^{Q,\ NUT} \geq \widetilde{Q}^{UT}$, $\forall_i \in O_i^{\ NUT}$ then task O_i should therefore be given higher priority and pushed in Oi ^{MT}. Dequeue (O_i^{NUT}) Q^{MT} i.e remove the task from the Non-Urgent queue and put it in Moderate Queue on an FCFS basis.

Rule 2: When $Q_3^{\ MT} = \varphi$ and $Q^{NUT} \neq \varphi$ then offload (O_i) to the cloud in FCFS order by dequeuing it with Priority O_i^{NUT}

Rule 3: When $Q_3^{MT} \neq \varphi$ and $Q^{NUT} = \varphi$ then offload (O_i) to the cloud in FCFS order by dequeuing it with Priority O_i^{MT}.

Algorithm 1. PaTS (Priority aware Task Scheduling)

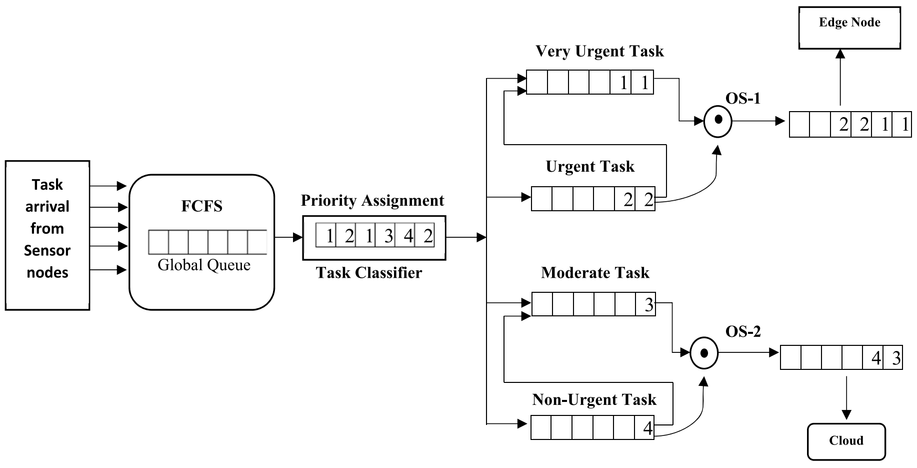


Fig. 2. An example of the Task Scheduling program on the gateway device.

```
Input:   O: Group of Task, Task attributes: $O_i^Q$ and $O_i^F$
Output: Incoming Tasks Ideal Schedule
    a. For $O_i \leftarrow$ 1 to O do
    b. Determine the Utilization Factor $U_i = O_i^Q / O_i^F$
    c. Give Priority to each task Oi based on UF.
    d. Each task Oi should be added to the Priority Queue Q.
    e. Assign a task to $Q^{VU}$, $Q^{U}$, $Q^{M}$ and $Q^{NU}$ using condition as
       $U_i \leq \frac{1}{4}$, $\frac{1}{4} < U_i < \frac{1}{3}$, $\frac{1}{3} < U_i < \frac{1}{2}$ and $U_i > \frac{1}{2}$
    f. End For
    g. For $O_i \leftarrow$ 1 to O do
    h. Begin Time Quantum $W_i^Q = 0$;
    i. Prioritize tasks $O_i$ and schedule them;
    j. After a certain Time Quantum, dequeue () task $O_i$.;
    k. Update $W_i^Q = W_i^Q + \widetilde{Q}$;

       End For.
```

Algorithm 2. NSGA-2 Pseudocode for Energy and Queue Delay Optimization.

```
Input:   Size of Population = N, Maximum Iterations = m, Problem: F(x)
Output: A collection of non-dominant solutions.
  1.  Suppose $t \leftarrow 0$
  2.  $P_t \leftarrow$ new population of size, n
  3.  $P_t \leftarrow$ Each objective function's worth is evaluated for the population $P_t$.
  4.  $[P_t, F] \leftarrow$ a fast $P_t$ population sorting using a non-dominated approach.
  5.  for t =1 to m do
  6.    for i =1 to n do
  7.      $s \leftarrow$ population ($P_t$)
  8.      $\eta_c$, $\eta_m \leftarrow$ Cross over and mutation strategy.
  9.      $O_i \leftarrow$ produce a population of children with a size of n using $\eta_c$, $\eta_m$.
  10.     $R_t \leftarrow P_t \cup O_i$  // generate a new population of size 2n
  11.     $F \leftarrow$ Fast non-dominated sorting of $R_t$
  12.     $P_{t+1} \leftarrow \emptyset$
  13.     $c \leftarrow 1$          //Initialization Counter
  14.     while $|P_{t+1}| + |F_i| \prec z$ do
  15.          $R_t(F_i) \leftarrow$ Crowding distance calculated $R_t(F_i)$
  16.          $P_{t+1} \leftarrow P_t \cup R_t(F_i)$
  17.          $c \leftarrow c + 1$
  18.     end while
  19.          $F_i \leftarrow$ 1 crowd sort ($F_i \prec_c$)
  20.          $P_{t+1} \leftarrow P_{t+1} \cup R_t(F_i[1: z-|P_{t+1}|])$
  21.          $O_{t+1} \leftarrow$ Tournament Selection ($P_{t+1}$)
  22.          $O_{t+1} \leftarrow$ crossover
  23.          $O_{t+1} \leftarrow$ mutation
  24.          $t \leftarrow t + 1$
  25.    end for
  26. end for
  27. End
```

In our work, we formulated a multi-objective problem i.e., to minimize energy and queue delays. To optimize the problem, we adopted a four-level queue model where separate pairs of queues are created for Urgent and Non-Urgent tasks. Furthermore, the task scheduling policy is framed to assign priority to every task and later tasks are enqueued into their respective queues. The efficiency of the proposed policy is evaluated using a Bio-Inspired multi-objective Genetic algorithm called NSGA-2 which is a very popular multi-objective optimization technique.

6. Experimental results

This section evaluates the effectiveness of the suggested PaTS approach, using the benchmark algorithm Energy Aware Scheduling (EaS) over different queueing performance metrics like Queuing delay, Computation Time and Energy consumption.

6.1. Simulation setup

To assess the effectiveness of the suggested Priority Aware Task Scheduling (PaTS) Policy, a Bio-inspired multi-objective NSGA-2 algorithm is used. It is one of the benchmark multi-objective Techniques to optimize two objectives of the PaTS algorithm, i.e., minimize energy and Queue Delay. The NSGA-2 code file will be made to run in MATLAB. The effectiveness of the suggested algorithm will be demonstrated by comparing the results of the simulation with benchmark results.

In our simulation, number of tasks O considered will be [100, 1000], data size in each task $O_i^{Z} = [0,10]$ in megabytes over a time interval of Δt [18]. Processing density O_i^{PR} of O_i^{th} task = 1900[cycles/byte]. For the classification of a task, upper and lower values are considered for task frequency $O_i^{F} = [1,10]$, CPU requirement $O_i^{CPU} = [0.1, \ldots, 0.5]$ GHz, execution deadline $O_i^{d} = [0, 10]$.The arrival rate of tasks $\lambda = [0.1, 0.2 \ldots, 0.9]$. Also, the number of sensor devices taken S = 200, Centralized Cloud Servers C = 2, Edge devices E = 20, bandwidth β = 20Mhz and Queues taken are Q = 4.

6.2. Results

6.2.1. Average queuing delay

Fig. 5(a) presents the queue delay of various priority queues by varying tasks. It is observed that queue delay for tasks set with four queues for various tasks is low compared with other queues. Fig. 5(b) depicts the comparative analysis of the average queue delay of the proposed method with the EaDO algorithm. The results show that the proposed technique has a lower average queueing delay than the baseline algorithms for a variety of priority-aware tasks.

6.2.2. Processing time

The processing time takes into account all aspects like processing of data, general processing, and getting output from remote machines. Our suggested PaTS technique offloads tasks which have low priority to the centralized cloud server while offloading high priority tasks to the nearby surrounding edge servers according to the priority. The total

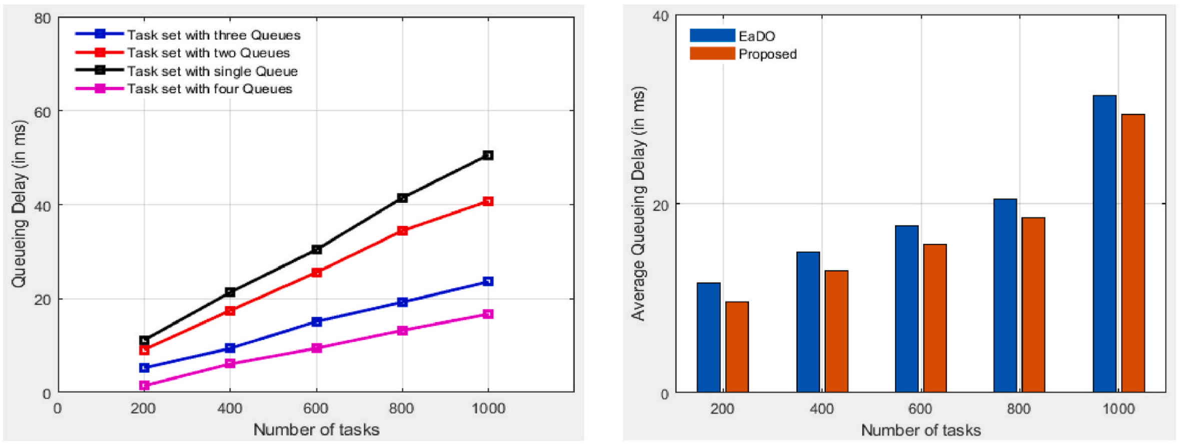


Fig. 5. Queue delay of (a) Different Priority Queues, (b) Compared to current works.

processing time for various priority jobs that met delay deadlines across various computing devices is shown in Fig. 6(a). Performance analysis of our proposed technique in comparison to recent works is shown in Fig. 6 (b). This investigation demonstrates that our suggested PaTS technique reduces delay by 10%–15% and is much more efficient for handling tasks which have high priority in IoT-based sensor networks.

6.2.3. Average energy usage

The data size (O_i^Z), bandwidth utilized, and f_j CPU of a computing device (V_j) have a significant impact on the sensor network energy as a parameter. Fig. 7(a), shows the overall energy consumed (E_{ij}^{Total}) by the processing devices during the computation of the tasks. As analyzed in Fig. 7(a), compared to Edge devices and cloud servers, the sensing device's total energy is lower as the processing power of the sensing device is limited as compared to the other two computing devices. Additionally, it can be seen in Fig. 7(b) that the proposed PaTS approach uses 15 mW of energy, while the existing benchmark algorithm EaDO consumes 17 mW of energy, which is greater by 11% over the proposed algorithm. It is observed that the total energy (E_{ij}^{Total}) used by Fog networks depends upon E_{ij}^U, E_{ij}^D, and E_{ij}^V, respectively. The T_{ij}^U and T_{ij}^D of the tasks will change due to an increase in distance between the IoT and computing devices, which also raises the task's E_{ij}^{Total}

7. Conclusion

This paper focused on the effects of task prioritization on industrial-based sensor networks. First, the priority of the incoming task is assigned and then enqueued into their respective queues. Second, the tasks enqueued in the respective queues are dequeued and offloaded to edge and cloud servers using the scheduled queue policy of the multilevel feedback queue model. When compared to benchmark algorithms in terms of average queue delay, computational time, and energy consumption, simulation results demonstrate the algorithm's effectiveness as shown in Table 3. In the future, a technique will be established to offload scheduled tasks onto a suitable computing device and then the entire framework will be compared with previous works to improve Quality of Service (QoS) requirements.

CRediT authorship contribution statement

Malvinder Singh Bali: Conceptualization, Data curation, Writing – original draft. **Kamali Gupta:** Conceptualization, Data curation, Writing – original draft. **Deepali Gupta:** Conceptualization, Data curation, Writing – original draft. **Gautam Srivastava:** Investigation, Formal analysis, Writing – original draft, Supervision. **Sapna Juneja:** Investigation, Writing – original draft. **Ali Nauman:** Supervision, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

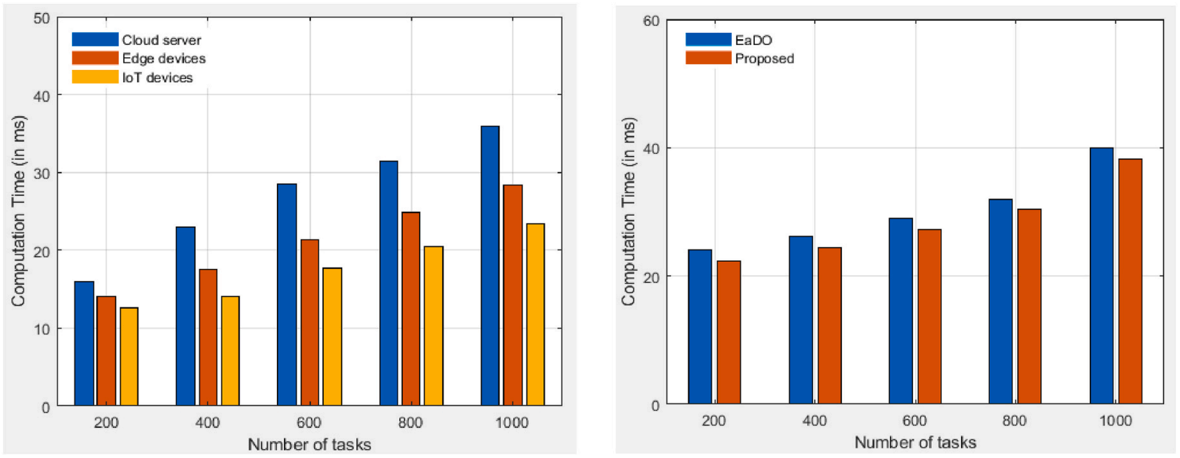


Fig. 6. Delay in the rate of consumption in (a) Various Processing devices, (b) Compared with recent works.

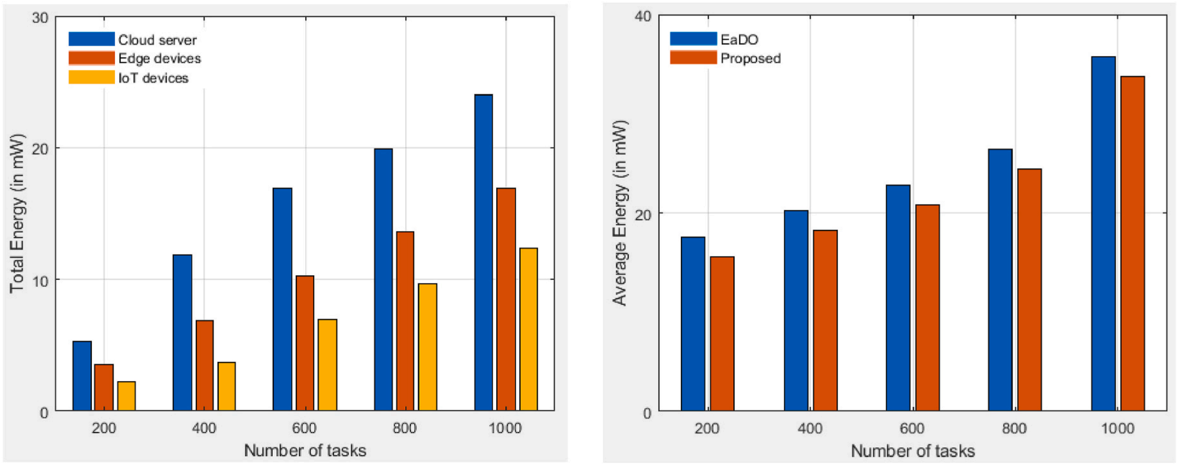


Fig. 7. Rate of Energy Usage: (a) Different Processing Devices, (b) Compared to previous works.

Table 3
Comparative analysis of results with the benchmark algorithm.

Number of tasks	Average Queue Delay			Computation Time			Average Energy		
	EaDO	Proposed	%age improvement	EaDO	Proposed	%age improvement	EaDO	Proposed	%age improvement
200	11.66	9.66	17.24	24	22.3	7.08	17.54	15.54	11.40
400	14.9	12.9	13.42	26.16	24.46	6.49	20.19	18.19	9.90
600	17.64	15.64	11.33	28.91	27.21	5.88	22.84	20.84	8.75
800	20.48	18.48	9.76	31.96	30.26	5.31	26.47	24.47	7.55
1000	31.43	29.43	6.36	39.95	38.25	4.25	35.78	33.78	5.58

Data availability

Data will be made available on request.

References

[1] Fuqoha Ala, Guzani Mohsen, Mohammadi Mehdi, Aledhari Mohammad, Ayyash Moussa, Internet of things: a survey on enabling technologies, Protocols and Applications, IEEE Commun. Surv. Tutorials 17 (4) (2015) 2347–2376, https://doi.org/10.1109/COMST.2015.2444095.

[2] Mohammad Hasan, State of IoT 2022: No of Connected IoT devices growing 18% to 14.4 billion globally, IoT-Analytics.com (2022, May 18). https://iot-analytics.com/number-connected-iot-devices.

[3] Aliasghar Azma, Nima Kianfar, Hassein Chitsazi, Research and development on cloud computing, in: 5th International Conference on Advance Research in Science Engineering and Technology, 2021.

[4] Hajime Kanzaki, Kevin Schubert, Nicholas Bambos, Video streaming schemes for industrial IoT, in: 26th International Conference on Computer Communications and Networks (ICCCN), 2017, https://doi.org/10.1109/ICCCN.2017.8038533.

[5] Shivanjali Khare, Michael Totaro, Big data in IoT, in: 10th International Conference on Computing, Communication and Networking Technology (ICCCNT), July 2019, https://doi.org/10.1109/ICCCNT45670.2019.8944495.

[6] Arkady Zaslowsky, Dimitrios Georgakopoulos, Chrith Perera, Sensing as a Service and big data, in: International Conference on Advances in Cloud Computing (ACC), July 2012.

[7] Weisong Shi, Pallis George, Zlvivei Xu, “Edge Comput.Proc.IEEE 107 (August 2019), https://doi.org/10.1109/JPROC.2019.2928287.

[8] Tie Qui, Jiancheng Chi, Xlaobo Zhou, Mohd Aliquzzaman Zhaolong, Dapeng Oliver Wu, Edge computing in industrial Internet of things: architecture, Adv.Chall. IEEE Commun. Surv. Tutorials 22 (4) (2020), https://doi.org/10.1109/COMST.2020.3009103.

[9] Md Sajjad, Cosmas Nwakanwa, Jae Lee, Kim Dong, Edge computational task offloading scheme using reinforcement learning for IIoT scenario, ICT Express 6 (4) (Dec 2020), https://doi.org/10.1016/j.icte.2020.06.002.

[10] You Qian, Bing Tang, Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial IoT, J. Cloud Comput. (July 2021), https://doi.org/10.1016/j.ifacol.2021.04.122.

[11] H. Wang, Q. Wang, Y. Li, G. Chen, Y. Tang, Application of fog architecture based on multi-agent mechanism in cpps, in: 2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2), 2018, pp. 1–6.

[12] N. Yoshikane, et al., First demonstration of geographically unconstrained control of an industrial robot by jointly employing sdnbased optical transport networks and edge compute, in: 2016 21st Opto Electronics and Communications Conference (OECC) Held Jointly with 2016 International Conference on Photonics in Switching (PS), 2016, pp. 1–3.

[13] I.A. Tsokalo, H. Wu, G.T. Nguyen, H. Salah, F.H.P. Fitzek, Mobile edge cloud for robot control services in industry automation, in: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2019, pp. 1–2.

[14] T.M. Jose, A novel sensor-based approach to predictive maintenance of machines by leveraging heterogeneous computing, in: 2018 IEEE SENSORS, 2018, pp. 1–4.

[15] L. Li, K. Ota, M. Dong, Deep learning for smart industry: efficient manufacture inspection system with fog computing, IEEE Trans. Ind. Inf. 14 (10) (2018) 4665–4673.

[16] H.-S. Lee, J.-W. Lee, Resource and task scheduling for SWIPT IoT systems with renewable energy sources, IEEE Internet Things J. 6 (2) (Apr. 2019) 2729–2748.

[17] Y. Yu, J. Zhang, K.B. Letaief, Joint subcarrier and CPU time allocation for mobile edge computing, in: Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2016, pp. 1–6.

[18] Abhishek Hazra, Mainak Adhikari, Tarachand Amgoth, Satish Narayana Srirama, Fog computing for energy-efficient data offloading of IoT applications in industrial sensor networks” in, IEEE Sensor. J. 22 (9) (May 2022), https://doi.org/10.1109/JSEN.2022.3157863.

[19] Shuran Sheng, Peng chen, Zhamin Chen, Lenan Wu, Yuxuan Yao, Deep reinforcement learning- based task scheduling in IoT edge computing, in: IEEE International Conference on Communication (ICC), June 2020, https://doi.org/10.1109/ICC40277.2020.9148888. Dublin.

[20] H. Tan, Z. Han, X. Li, F.C.M. Lau, Online job dispatching and scheduling in edge-clouds, in: Proceedings of the IEEE INFOCOM 2017—IEEE Conference on Computer Communications, May 2017, pp. 1–9. Atlanta, GA, USA, 1–4.

[21] Y. Zhang, P. Du, J. Wang, T. Ba, R. Ding, N. Xin, Resource scheduling for delay minimization in multi-server cellular edge computing systems, IEEE Access 7 (2019) 86265–86273.

[22] X. Chen, N. Thomas, T. Zhan, J. Ding, A hybrid task scheduling scheme for heterogeneous vehicular edge systems, IEEE Access 7 (2019) 117088–117099.

[23] C. Li, J. Tang, H. Tang, Y. Luo, Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment, Future Generat. Comput. Syst. 95 (2019) 249–264.

[24] H.A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, C. Assi, Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing, IEEE J. Sel. Area. Commun. 37 (2019) 668–682.

[25] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A.K. Bashir, A. Jolfaei, N. Kumar, Energy-aware marine Predators algorithm for task scheduling in IoT-based fog computing applications, IEEE Trans. Ind. Inf. (2020) (early access).

[26] J. Meng, H. Tan, X. Li, Z. Han, B. Li, Online deadline-aware task dispatching and scheduling in edge computing, IEEE Trans. Parallel Distr. Syst. 31 (2020) 1270–1286.

[27] M. Molina, O. Munoz, A. Pascual-Iserte, J. Vidal, Joint scheduling of communication and computation resources in multiuser wireless application offloading, Proc. IEEE PIMRC (Sep. 2014) 1093–1098.

[28] Y. Sahni, J. Cao, L. Yang, Data-aware task allocation for achieving low latency in collaborative edge computing, IEEE Internet Things J. 6 (2) (Apr. 2019) 3512–3524.

[29] Y. Sahni, J. Cao, L. Yang, Data-aware task allocation for achieving low latency in collaborative edge computing, IEEE Internet Things J. 6 (2) (Apr. 2019) 3512–3524.

[30] Y. Kim, C. Song, H. Han, H. Jung, S. Kang, Collaborative task scheduling for IoT-assisted edge computing, IEEE Access J. (Nov. 2020), https://doi.org/10.1109/ACCESS.2020.3041872.

[31] Q. Ju, G. Sun, H. Li, Y. Zhang, Collaborative in-network processing for Internet of things of battery less things, Internet of Things J (2019), https://doi.org/10.1109/JIOT.2019.2899022.

[32] Malvinder singh Bali, Kamali Gupta, shalli Rani, Rajnish Ratna, An energy -efficient partial data offloading- based priority rate controller technique in edge based IoT network to improve the QoS, Wireless Commun. Mobile Comput. (2022), https://doi.org/10.1155/2022/4288663.

[33] Malvinder bali, Kamali Gupta, Deepika Koundal, Atef Zaguia, Shubham Mahajan, Amit Kant Pandit, Smart architectural framework for symmetrical data offloading in IoT, Symmetry J (2021), https://doi.org/10.3390/sym13101889.

[34] Kanwal preet Kour, Deepali Gupta, Kamali Gupta, Gaurav Dhiman, Sapna Juneja, Wattana Viriyasitavat, Hamidraza Mohafez, Mohammad AminulIslam, Smart-hydroponic-based framework for saffron cultivation: a precision smart agriculture perspective, Sustainability J (2022), https://doi.org/10.3390/su14031120.

[35] M. Adhikari, M. Mukherjee, S.N. Srirama, Dpto: a Deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing, IEEE Internet Things J. 7 (7) (2020) 5773–5782.

[36] X. Xu, B. Shen, S. Ding, G. Srivastava, M. Bilal, M.R. Khosravi, V.G. Menon, M. A. Jan, W. Maoli, Service offloading with deep Q-network for digital twinning empowered Internet of vehicles in edge computing, IEEE Trans. Ind. Inf. (2020), 1–1.

[37] Q. Wang, S. Guo, J. Liu, Y. Yang, Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing, Sustain.Comput.: Inf. Syst. (2019), https://doi.org/10.1016/j.suscom.2019.01.007.

[38] Sandhya Sharma, et al., Recognition of gurmukhi handwritten city names using deep learning and cloud computing, Sci. Program.2022 (2022).

[39] S. Kanwal, J. Rashid, J. Kim, G. Dhiman, A. Hussain, Mitigating the coexistence technique in wireless body area networks by using superframe interleaving, IETE J. Res. (2022) 1–15.

[40] S. Mittal, A. Bansal, H. Turabieh, M.M. Elarabawy, Z.K. Bitsue, Using identity-based cryptography as a foundation for an effective and secure cloud model for E-health, Comput. Intell. Neurosci. (2022).

[41] N. Gupta, K. Gupta, D. Gupta, S. Juneja, H. Turabieh, G. Dhiman, W. Viriyasitavat, Enhanced virtualization-based dynamic bin-packing optimized energy management solution for heterogeneous clouds, Mathematical Problems in Engineering, 2022 (2022).

[42] S. Sharma, H. Turabieh, S. Sharma, SWOT: a hybrid hardware-based approach for robust fault-tolerant framework in a smart day care, Secur. Commun. Network.2022 (2022).

[43] M. Uppal, A. Sulaiman, K. Rajab, A. Rajab, A. Shaikh, Cloud-based fault prediction for real-time monitoring of sensor data in hospital environment using machine learning, Sustainability 14 (18) (2022), 11667.

[44] H.K. Upadhyay, G. Muhammad, A. Nauman, N.A. Awad, Analysis of IoT-related ergonomics-based healthcare issues using analytic hierarchy process methodology, Sensors 22 (21) (2022) 8232.

[45] M. Li, N. Mao, X. Zheng, T.R. Gadekallu, Computation offloading in edge computing based on deep reinforcement learning, in: A.K. Bashir, G. Fortino, A. Khanna, D. Gupta (Eds.), Proceedings of International Conference on Computing and Communication Networks. Lecture Notes in Networks and Systems, vol. 394, Springer, Singapore, 2022, https://doi.org/10.1007/978-981-19-0604-6_28.

[46] T.R. Gadekallu, Q.V. Pham, D.C. Nguyen, P.K.R. Maddikunta, N. Deepa, B. Prabadevi, W.J. Hwang, Blockchain for edge of things: applications, opportunities, and challenges, IEEE Internet Things J. 9 (2) (2021) 964–988.

[47] O. Nafea, W. Abdul, G. Muhammad, and M. Alsulaiman, "Sensor-based human activity recognition with spatio-temporal deep learning," Sensors, vol. 21, no. 6, Article ID: 2141, March 2021. DOI: 10.3390/s21062141.

[48] H. Altaheri, G. Muhammad, M. Alsulaiman, et al., Deep learning techniques for classification of electroencephalogram (eeg) motor imagery (mi) signals: a review, Neural Comput. Appl. (2022), https://doi.org/10.1007/s00521-021-06352-5.

[49] Ravi Gatti, G.B. Arjun Kumar, K.N. Sunil Kumar, Satyasrikanth Palle, Thippa Reddy Gadekallu, Optimal resource scheduling algorithm for cell boundaries users in heterogenous 5G networks, Phys. Commun. 55 (2022).