Federated Learning for Cost Optimized Offloading in Edge-enabled Industrial Internet of Things

Abhishek Hazra[*$], Bhabesh Mali[+], Alakesh Kalita[*], Mohan Gurusamy[*]
[*]*National University of Singapore, Singapore, 119077*
[$]Indian Institute of Information Technology Sri City, India, 517646
[+]*Indian Institute of Technology Guwahati, India, 781039*
a.hazra@ieee.org, bhabeshmali43@gmail.com, alakesh@nus.edu.sg, gmohan@nus.edu.sg

***Abstract*—Industrial Edge Computing (IEC) is a viable solution for realizing tasks generated by the Industrial Internet of Things (IIoT) at the network's edge. Although resource optimization can be done at IEC, it is also crucial to reduce the overall execution cost and maintain the trade-off between energy consumption and service delay as these metrics are critical in the current industrial automation process and subsequent upcoming industrial revolutions. Considering these requirements, in this work, we design a cost-optimized framework for efficient task offloading in IIoT, specifically focusing on task classification, device selection, and task offloading issues. Our proposed framework named `FedOff` aims to concurrently manage industrial dynamic service requests and IEC resources simultaneously. To achieve this, we formulate an objective function as an energy-delay cost optimization problem for industrial networks. At first, we develop a heuristic task categorization technique called `De-Sparse` to select and execute delay-sensitive industrial tasks on networks' edge. Subsequently, we leverage federated learning to identify the most suitable IEC servers for task offloading by analyzing the historical non-Independent and Identically Distributed (IID) dataset generated by the IIoT devices. Experimental results show that our proposed `FedOff` strategy is cost-efficient and exhibits shorter delays in IIoT networks compared to the existing methods.**

***Index Terms*—Industrial Edge Computing, Task Offloading, Federated Learning, IIoT, Cost Optimization.**

I. Introduction

The Industrial Internet of Things (IIoT) has attracted significant attention from researchers and developers due to its intelligent and customer-centric task execution capabilities. Various applications such as coal mining, smart logistics, predictive maintenance, and sustainable grids generate vast amounts of data. This, in turn, presents challenges related to industrial operational costs in terms of energy consumption and transmission delays in edge networks [1]. With the advancement of Industrial Edge Computing (IEC), delay-sensitive IIoT applications can now be processed at the network edge. Thus, energy consumption and latency can be reduced significantly. A new generation of edge networks based on Artificial Intelligence (AI) enabled technology can effectively and efficiently handle massive industrial applications through intelligent task offloading techniques. Thus, task classification, device selection, and computation offloading strategies are crucial to developing an efficient, autonomous IIoT framework to optimize end-to-end computation costs.

Due to the limited computational capacity of IEC servers, industrial applications can employ offloading techniques to maximize their resource utilization capacity. Federated Learning (FL)-enabled technologies have recently been used as the foundation for intelligent task offloading and deploying vast industrial applications using collaborative intelligence and knowledge sharing [2]. However, the limited capability of the IEC server cannot be neglected, and the deployment of IIoT applications in industrial networks still encounters significant obstacles. Despite the fact that *Cloud* has the potential to handle storage requirements and huge data processing, delay-critical industrial applications should address latency and energy consumption issues. For such applications, task offloading through FL would play a crucial role in facilitating network training on local IEC servers (edge servers located in industrial networks) instead of using a central server. It has also been shown in the literature that FL optimizes offloading tasks to remote computing servers while preserving the integrity of industrial data. FL also provides flexibility to train a new network and construct a robust model in a time-critical and sensitive environment, which makes it completely different from other techniques.

A. Motivation

In a typical industrial environment, IIoT devices with limited resources offload computation data to centralized cloud servers or nearby IEC servers. Although IIoT devices, such as vision cameras, sensors, and robots, have a high data generation capacity, industrial edge networks demonstrate the ability to handle such data [3]. It is necessary that industrial devices consciously execute computationally intensive and delay-responsive applications depending on their execution demands. Therefore, an optimized computation offloading framework for industrial applications is necessary, where delay-sensitive IIoT applications are processed by both IIoT devices and IEC computing servers simultaneously within the energy-delay limit. However, designing such an optimized computation offloading framework in complex industrial networks is not trivial. Based on the current research gap, we propose probabilistic methods for task classification and FL techniques for efficient task offloading decisions that optimize costs. Therefore, the aim of this study is to develop a framework for offloading intelligent

2023 IEEE Future Networks World Forum (FNWF) | 979-8-3503-2458-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/FNWF58287.2023.10520588

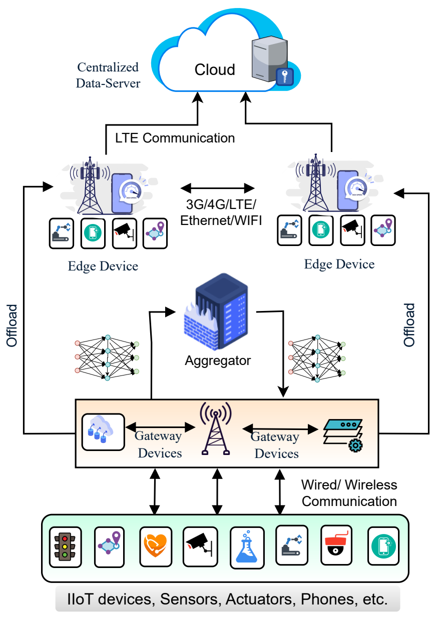


Fig. 1: Illustration of the industrial edge networks.

computations for IIoT applications in order to reduce the cost of computing while, at the same time, using minimum energy consumption and delays.

B. Related Work

In recent years, FL has been used in industrial edge networks in order to efficiently offload different tasks so that IIoT applications can be operated within their delay and energy constraints. For example, in [2], Hazra *et al.* have designed a computation offloading strategy that efficiently utilizes the resources of IEC servers in industrial networks. The work in [4] have designed a scheduling and computation offloading strategy for delay-sensitive IIoT applications by considering their processing deadlines. However, due to the limited resource capacity of the IEC servers, they can handle limited service requests out of the total assigned service requests. Similarly, the work in [5] used heuristic or greedy-based methods for service request selection and execution. Furthermore, cloud computing is not an efficient solution to handle delay-sensitive IIoT applications. Therefore, gathering knowledge about suitable computing devices and distributing tasks among IEC servers is preferable, which could be a much better approach to meet the requirements of such IIoT applications. Even though existing works such as [6] and [7] have analysed the ML-based data transmission strategy for IoT applications, these works do not consider the intelligent task offloading mechanism for edge networks.

Additionally, very few studies have studied FL methods for enhancing the capabilities of industrial edge networks [8]. Most research focuses on improving service quality aspects, such as response time, latency, and cost. Additionally, there is a lack of research on industrial edge networks that investigates task classification, device selection, collaborative intelligence, and compute offloading. While optimizing the network at a high level helps in reducing network delays and congestion, there is still room for improvement in overall data processing latency. In an industrial setting, FL approaches can facilitate information sharing, develop self-learning and self-healing capabilities within industrial networks, and promote collaboration among devices. FL-based techniques are particularly effective in developing a decentralized system as they can analyze untapped datasets in a coordinated manner. Therefore, the primary research objective of this paper is to design an intelligent computation offloading framework for time-critical industrial edge networks.

C. Contribution

Considering the above-mentioned challenges, in this work, we design a task-offloading framework that can make intelligent offloading decisions and optimize execution costs in industrial networks. First, we formulate a cost optimization problem with the objective of maximizing efficiency and minimizing energy delay in IEC networks. Next, we design a probabilistic `De-Sparse` task identification strategy to classify IIoT tasks into local (*i.e.,* IIoT executable) and remote (*i.e.,* IEC executable). Finally, we develop an FL model for robust IEC server selection and cost-optimized offloading. In brief, the major contributions of this work are as follows.

- We develop a novel task offloading strategy called `FedOff` to initially classify the IIoT generated task into local and remote executable and later offload the remote executable task to the most deserving IEC server.
- Based on the QoS constraints, such as energy and latency, we formulate a cost optimization problem with the objective of maximizing performance and minimizing energy and delay for IEC networks.
- To obtain near-optimal outcomes, we first classify the IIoT-generated task using our novel probabilistic `De-Sparse` function. Subsequently, we develop an FL model, intricately fed with the unique characteristics of the IIoT's non-IID data, to perform robust IEC server selection and offloading.
- With over 1000 independent simulation runs, we have validated that the proposed `FedOff` strategy can optimize overall execution costs while keeping energy consumption low and balancing delay.

The remainder of the paper is organized as follows. Section II defines the system model and problem formulation. Section III defines our proposed methodology. The performance analysis of our proposed strategy is illustrated in Section IV. Finally, Section V concludes our discussion.

II. System Model and Problem Formulation

Consider an edge-enabled IIoT network with a set of I IIoT-enabled devices $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_I\}$ and a set of J IEC servers $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_J\}$ located in an industrial environment. As illustrated in Fig. 1, IIoT devices generate number of tasks and each task can be defined by *3-tuple*, $\mathcal{Z}_i = \langle \mathcal{Z}_i^{\text{IN}}, \mathcal{Z}_i^{\text{CPU}}, \mathcal{Z}_i^{\text{MM}} \rangle$, where $\mathcal{Z}_i^{\text{IN}}$ represents the task size, $\mathcal{Z}_i^{\text{CPU}}$ defines the CPU requirement, and $\mathcal{Z}_i^{\text{MM}}$ indicates the execution deadline. In addition, a binary task offloading decision vector $\mathcal{K}(i,j) \in \{0,1\}$ is used to determine whether tasks can be executed locally or offloaded to other high-capacity IEC servers, defined as follows.

$$\mathcal{K}(i,j) = \begin{cases} 1, & \text{if } i\text{th task executed on } j\text{th IEC server} \\ 0, & \text{otherwise local IIoT devices.} \end{cases}$$

To make the system model more functional, we consider IIoT devices can request multiple computing devices. However, the task execution process will be done on one computing device.

A. IIoT Execution Model

Initially, industrial devices try to execute their tasks on devices, and a task is allowed to execute on IIoT devices if and only if $\mathcal{Z}_i^{\text{CPU}} < \mathcal{Z}_i^{\text{M\_CPU}}$, where $\mathcal{Z}_i^{\text{M\_CPU}}$ is a CPU constant. Let $\mathscr{C}_i^{\text{CPU}}$ denote the computation capacity of the IIoT devices. Then we can define the local execution delay $\mathcal{O}_i^{\text{IIoT}}$ and corresponding energy consumption $\mathcal{X}_i^{\text{IIoT}}$ rate for executing a task on the IIoT device as follows.

$$\mathcal{O}_i^{\text{IIoT}} = \mathcal{K}(i,j)\mathcal{Z}_i^{\text{CPU}}\mathcal{Z}_i^{\text{IN}}(\mathscr{C}_i^{\text{CPU}})^{-1} \tag{1}$$

$$\mathcal{X}_i^{\text{IIoT}} = \mathcal{K}(i,j)\mathcal{Z}_i^{\text{CPU}}\mathcal{Z}_i^{\text{IN}}k\big(\mathscr{C}_i^{\text{CPU}}\big)^2 \tag{2}$$

We assume that energy dissipation for IIoT devices follows $k\big(\mathscr{C}_i^{\text{CPU}}\big)^3$ energy model with k chip coefficient [6].

B. Edge Server Execution Model

Due to the confined storage and processing capabilities, IIoT devices prefer to offload excess workloads to nearby computing devices. Let γ_i and $\zeta_{i,j}$ represent the data transmission power and channel gain of an IIoT device $\mathcal{A}_i$. Then we can define the transmission rate with $\mathcal{B}_i \geq 0$. The available link capacity can be represented as $\widetilde{\mathcal{X}}_{i,j} = \mathcal{B}_i log_2\left(1 + \frac{\gamma_i \zeta_{i,j}}{\mathscr{A}_i}\right)$, where $\mathscr{A}_i$ represents the transmission noise over the communication channel. Then the offloading time $\mathcal{O}_i^{\text{upload}}$ and corresponding offloading energy consumption $\mathcal{X}_i^{\text{upload}}$ to remote computing servers $\mathcal{H}_j$ can be represented as.

$$\mathcal{O}_i^{\text{upload}} = \frac{\mathcal{K}(i,j)\mathcal{Z}_i^{\text{IN}}}{\widetilde{\mathcal{X}}_{i,j}} \tag{3}$$

$$\mathcal{X}_i^{\text{upload}} = \frac{\mathcal{K}(i,j)\mathcal{Z}_i^{\text{IN}}}{\widetilde{\mathcal{X}}_{i,j}}\gamma_i \tag{4}$$

Specifically, IIoT devices offload more latency-critical tasks to nearby IEC servers and the remaining tasks to the cloud servers. Let $\mathscr{C}_j^{\text{IEC}}$ denote the CPU frequency of a computing device. Then the processing delay $\mathcal{O}_i^{\text{process}}$ and energy consumption $\mathcal{X}_i^{\text{process}}$ for executing a task on the computing device j can be defined as.

$$\mathcal{O}_i^{\text{process}} = \mathcal{K}(i,j)\mathcal{Z}_i^{\text{CPU}}\mathcal{Z}_i^{\text{IN}}(\mathscr{C}_j^{\text{IEC}})^{-1} \tag{5}$$

$$\mathcal{X}_i^{\text{process}} = \mathcal{K}(i,j)\mathcal{Z}_i^{\text{CPU}}\mathcal{Z}_i^{\text{IN}}k(\mathscr{C}_j^{\text{IEC}})^2 \tag{6}$$

In this system model, we consider remote task execution as the combination of uploading and processing data rates. As the downloading rate is only the 5% of offloading rate, we simply skipped this calculation. Thus, the task offloading delay $\mathcal{O}_i^{\text{IEC}}$ and energy consumption $\mathcal{X}_i^{\text{IEC}}$ on a remote computing device can be defined as follows.

$$\mathcal{O}_i^{\text{IEC}} = \frac{\mathcal{K}(i,j)\mathcal{Z}_i^{\text{IN}}}{\widetilde{\mathcal{X}}_{i,j}} + \frac{\mathcal{K}(i,j)\mathcal{Z}_i^{\text{CPU}}\mathcal{Z}_i^{\text{IN}}}{\mathscr{C}_j^{\text{IEC}}} \tag{7}$$

$$\mathcal{X}_i^{\text{IEC}} = \frac{\mathcal{K}(i,j)\mathcal{Z}_i^{\text{IN}}}{\widetilde{\mathcal{X}}_{i,j}}\gamma_i + \mathcal{K}(i,j)\mathcal{Z}_i^{\text{CPU}}\mathcal{Z}_i^{\text{IN}}k(\mathscr{C}_j^{\text{IEC}})^2 \tag{8}$$

C. Problem Formulation

The primary objective is to optimize the weighted energy-delay rate for tasks generated within edge-enabled industrial networks by IIoT. To accomplish this goal, we formulate precise calculations for assessing energy consumption and delay pertaining to task execution on both IIoT and remote computing devices. These calculations are defined as follows.

$$\mathcal{O}^{\text{total}}(\mathcal{K},\mathscr{C}) = \mathcal{K}(i,j)\mathcal{O}^{\text{IIoT}} + \mathcal{K}(i,j)\mathcal{O}^{\text{IEC}}(\mathcal{K},\mathscr{C}) \tag{9}$$

$$\mathcal{X}^{\text{total}}(\mathcal{K},\mathscr{C}) = \mathcal{K}(i,j)\mathcal{X}^{\text{IIoT}} + \mathcal{K}(i,j)\mathcal{X}^{\text{IEC}}(\mathcal{K},\mathscr{C}) \tag{10}$$

Therefore, given a set of IIoT devices, computing servers, offloading decision vector and computation capacity of computing devices, we can derive our objective function as follows.

$$\mathcal{J}(\mathcal{K},\mathscr{C}) = \sum_{i\in I}\Big\{\mu\ \mathcal{X}_i^{\text{total}}(\mathcal{K},\mathscr{C}) + \eta\ \mathcal{O}_i^{\text{total}}(\mathcal{K},\mathscr{C})\Big\} \tag{11}$$

where $\mu, \eta \in [0,1]$. Mathematically, the objective function and associated constraints can be defined as follows.

$$\begin{aligned} \underset{\mathcal{K},\mathscr{C}}{\text{minimize}} \quad & \mathcal{J}(\mathcal{K},\mathscr{C}) && \text{(12a)} \\ \text{subject to} \quad & \mathcal{K}(i,j) \in \{0,1\},\ \forall i \in \mathcal{A}, && \text{(12b)} \\ & \mathcal{B}_i \geq 0,\ \forall i \in \mathcal{A}, && \text{(12c)} \\ & \sum_{i\in\mathcal{A}} \mathcal{K}(i,j)\mathscr{C}_i \leq \mathscr{C}^{\text{MAX}}, && \text{(12d)} \\ & \sum_{i=1}^{|\mathcal{A}|}\sum_{j=1}^{|\mathcal{H}|} \mathcal{K}(i,j) \leq |\mathcal{H}|, && \text{(12e)} \\ & \sum_{i=1}^{|\mathcal{A}|} \mathcal{K}(i,j) = 1 && \text{(12f)} \end{aligned}$$

where $\mathcal{K} = \{\mathcal{K}(i,j)|\ \forall i \in \mathcal{A}\}$ and $\mathscr{C} = \{\mathscr{C}_j^{\text{IEC}}|\ \forall j \in \mathcal{H}\}$. Constraint (12b) pertains to the binary decision profile for task offloading, determining whether a task is executed locally or offloaded to remote servers. Constraint (12c) imposes restrictions on the availability of transmission bandwidth, ensuring that the network can handle the data transfer requirements. Constraint (12d) limits the total computation requests processed by edge servers, preventing overloading situations beyond a predefined threshold. Constraint (12e) sets a bound on the execution delay, ensuring that the time required for task completion remains within an acceptable threshold. Finally, Constraint (12f) allows for a one-device execution policy, enabling tasks to be executed on a single designated device.

III. Proposed Methodology

This section presents our proposed cost-optimized task offloading strategy in industrial edge networks. In the objective function, we can observe that the problem is NP-hard and finding the optimal solution to the problem is extremely difficult [9]. To overcome this issue, we have proposed the `FedOff` strategy, combining heuristic and FL techniques for making the best suitable decision. First, we introduce the novel task classification technique `De-Sparse`. Next, we develop an FL model using non-IID IIoT-generated data to make reliable and trustworthy device selection decisions.

A. Task Classification

We have introduced a novel task classification strategy called `De-Sparse` to differentiate IIoT-generated tasks into IIoT and IEC executable tasks. Specifically, `De-Sparse` restricts the IIoT executable tasks to a certain level, so excessive workloads can be offloaded to nearby IEC servers for execution. Let $\mathscr{E} = \{\mathscr{E}_1, \mathscr{E}_2, \ldots, \mathscr{E}_i\}$ denotes the input fed into the `De-Sparse` function. Each value in $\mathscr{E}$ is composed of a ratio between $\mathcal{Z}_i^{\text{CPU}}$ and $\mathscr{C}_i^{\text{CPU}}$. Initially, we invert the ordered array $\mathscr{E}$ and retain its values in $\mathscr{E}^{\text{SORT}}$. Subsequently, we calculate the mean value of the sorted array and store it in $\mathscr{E}^{\text{MEAN}}$. Next, we generate two more arrays $a^{\text{ARR}} = 1 + a_i \times \mathscr{E}_i^{\text{SORT}}$ and $\mathscr{E}^{\text{DIFF}} = |\mathscr{E}^{\text{MEAN}} - \mathscr{E}^{\text{SORT}}|$, where $a = [0 \text{ to } (\mathcal{L}(\mathscr{E}) - 1)]$. A boolean array $\mathscr{E}^{\text{BOOL}} = a^{\text{ARR}} > \mathscr{E}^{\text{DIFF}}$ is also devised to maintain a sequence among the tasks. Finally, $a^{\text{MAX}} = \max([i \mid i \in 0, 1, \ldots, i-1, \mathscr{E}_i^{\text{BOOL}} = \text{True}]) + 1$ is calculated to define $\lambda^{\mathscr{E}} = \frac{\mathscr{E}^{\text{DIFF}}[a^{\text{MAX}}] - 1}{a^{\text{MAX}}}$. Now, we can define the `De-Sparse` probability $\mathscr{E}^{\text{SP}}$ for the set of industrial tasks.

$$\mathscr{E}^{\text{SP}} = \max\ (\mathscr{E} - \lambda^{\mathscr{E}}, 0) \tag{13}$$

By leveraging the attributes of $\mathscr{E}^{\text{SP}}$ and threshold ψ, we gain the capacity to differentiate and classify tasks as either IIoT or IEC server executable.

- **IIoT Executable:** *The classification of a task set as IIoT executable can be established when* $\mathscr{E}^{\text{IIoT}} = [i \mid i \in 0, 1, \ldots, i-1, \mathscr{E}_i^{\text{SP}} \leq \psi]$.
- **IEC Executable:** *The classification of a task set as IEC executable can be established when* $\mathscr{E}^{\text{IIoT}} = [i \mid i \in 0, 1, \ldots, i-1, \mathscr{E}_i^{\text{SP}} > \psi]$.

Algorithm 1 illustrates the complete steps of the task classification strategy.

Algorithm 1: `De-Sparse` algorithm

```
1 INPUT: $\mathcal{Z}_i^{\text{CPU}}$, $\mathscr{C}_i^{\text{CPU}}$
2 OUTPUT: IoT or IEC executable tasks
   1: Intialize $\mathscr{E}$ such that $\mathscr{E}_i = \frac{\mathcal{Z}_i^{\text{CPU}}}{\mathscr{C}_i^{\text{CPU}}}$,
   2: Perform $\mathscr{E}^{SORT} = rSort(\mathscr{E})$ and find $\mathscr{E}^{MEAN} = SOE(\mathscr{E}^{SORT})/\mathcal{L}(\mathscr{E}^{SORT})$
   3: Generate $a^{ARR} = 1 + a_i \times \mathscr{E}_i^{SORT}$ and
      $\mathscr{E}^{DIFF} = |\mathscr{E}^{\text{MEAN}} - \mathscr{E}^{\text{SORT}}|$, where
      $a = [0 \text{ to } (\mathcal{L}(\mathscr{E}) - 1)]$
   4: Calculate $\mathscr{E}^{BOOL} = a^{ARR} > \mathscr{E}^{DIFF}$
   5: Calculate $\lambda^{\mathscr{E}} = \frac{\mathscr{E}^{\text{DIFF}}[a^{\text{MAX}}] - 1}{a^{\text{MAX}}}$ such that
      $a^{\text{MAX}} = \max([i \mid i \in 0, 1, \ldots i-1, \mathscr{E}_i^{\text{BOOL}} = \text{True}]) + 1$,
   6: Find $\mathscr{E}^{\text{SP}} = \max(\mathscr{E} - \lambda^{\mathscr{E}}, 0)$,
   7: if ($\mathscr{E}_i^{\text{SP}} > \psi$) then
   8:     Categorize as IEC executable $\mathscr{E}_i^{\text{IEC}} = \mathscr{E}_i^{\text{SP}}$
   9: else if ($\mathscr{E}_i^{\text{SP}} \leq \psi$) then
  10:     Categorize as IIoT executable $\mathscr{E}_i^{\text{IIoT}} = \mathscr{E}_i^{\text{SP}}$
  11: end if
```

B. FL-based Cost-optimized Task Offloading

To maximize the efficiency of IEC networks while reducing energy-delay requirements, we have proposed a robust FL model, trained using the non-IID data generated from IIoT devices, to perform the optimal selection of edge nodes. As in traditional machine learning, where the data is centralized after being taken from each data owner, FL provides the flexibility to train a model in a decentralized fashion, *i.e.,* the data never leaves its locality [10]. FL not only equips us with the ability to construct a robust model but also ensures the preservation of data privacy from malicious users/organizations.

The inception of the device selection model commenced with the preparation of non-IID task offloading supervised data for every individual IIoT device, each assuming the role of a local client. A third-party server, say Ω, has been introduced, which will serve as the central aggregator to aggregate the weights of locally trained models, preparing a global model. Let Ω generate an initial model $\mathcal{M}_0$ with initial weights as $\mathcal{W}_0$. The model $\mathcal{M}_0$ will be distributed to each IIoT during a global communication round. The model underwent a comprehensive training process comprising G global communication rounds, during which the local models underwent F local epochs per communication round. Finally, to develop the global model during each communication round, we aggregated the weights of the network as follows.

$$\mathcal{W}_{k+1} = \sum_{m=1}^{\mathcal{I}} \frac{x_i}{x} \mathcal{W}_k^{\mathcal{A}_i} \tag{14}$$

where, $\mathcal{W}_{k+1}$ is the updated weights, $\mathcal{W}_k^{\mathcal{A}_i}$ is the previous weights, x_i represents the total number of samples in the ith IIoT device, and x represents the total number of samples. To expedite the process of choosing an IEC server for each task associated with a specific IIoT at the Gth round, the model's weights are stored in $\mathcal{MODEL}$. The complete `FedOff` training model has been illustrated in Algorithm 2.

IV. Numerical Analysis

This section briefly analyses the performance of our proposed `FedOff` task offloading strategy. Initially, we outline the various libraries and hardware requirements to carry out the simulation process for `FedOff` algorithm. Then, we detailed the process to generate and distribute the non-IID datasets among different IIoTs. The performance matrices considered to evaluate the proposed model are execution delay, energy consumption rate and overall system cost. Further, a comprehensive analysis with standard algorithms has been carried

Algorithm 2: `FedOff` algorithm

```
1 INPUT: $\mathcal{M}_o$, $x_i$, $\mathcal{S}$, $\mathcal{W}$
2 OUTPUT: Task offloading decisions
  1: Training samples $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_I\}$
  2: Let $\Omega$ be a third-party aggregator with initial weights $\mathscr{W}_0$
  3: for each $round$ in $G$ do
  4:     for each $\mathcal{A}_i$ in $\mathcal{A}$ do
  5:         $\mathcal{W}_{k+1}^{\mathcal{A}_i}$ = Local Execution $(\mathcal{A}_i, \mathcal{W}_k)$
  6:     end for
  7:     Aggregate $\mathcal{W}_{k+1} = \sum_{m=1}^{\mathcal{I}} \frac{x_i}{x} \mathcal{W}_k^{\mathcal{A}_i}$
  8: end for
  9: for each $epoch$ in $F$ do
 10:     Calculate loss and update weights
 11: end for
 12: Classify tasks using the DE-Sparse algorithm.
 13: Call weights of the FL model and store in $\mathcal{MODEL}$
 14: for each $\mathcal{A}_i$ in $\mathcal{A}$ do
 15:     for each $(IECTask)_j$ in $\mathcal{A}_i$ do
 16:         Offloading decision for $(IECTask)_j$ = $\mathcal{MODEL}(IECTask)_j$
 17:     end for
 18: end for
```

out to show the effectiveness of `FedOff` algorithm compared to various baseline methods such as `Wang's` strategy [11], `Mao's` strategy [12] and `Din's` strategy [13].

A. Simulation Setup

The simulation process has been carried out on an Intel Core-i5 CPU with 8GB of RAM and an Ubuntu operating system. We set $\psi = 0.01$ for the task classification. To simulate the FL model on the respective non-IID dataset, a total of 200 global communication rounds were considered, and for each round, the local model was trained for 10 epochs. A total of 3 IEC servers and 10 IIoT devices were taken to test our proposed approach. Other energy-, network- and computation capacity-related parameters for IIoT devices and IEC servers are considered from [4] and [6]. The FL process was simulated using the Flower Framework.

B. Federated Data Generation

The commencement of the global device selection model's development through FL training involved preparing input data for each IIoT. Given that the connection between each IIoT and IEC server is not uniform, the dataset exhibits distinct non-IID characteristics. This disparity arises due to varying output labels for each IIoT, resulting in observable variations among them. The dataset used for training the FL model was meticulously developed using the loose offloading decisions technique [14]. Each offloading decision is evaluated based on energy consumption and latency costs. The optimal minimum value is determined and subsequently recorded as an input sample within the dataset. Distinct datasets corresponding to each IIoT device, interconnected with their respective IEC

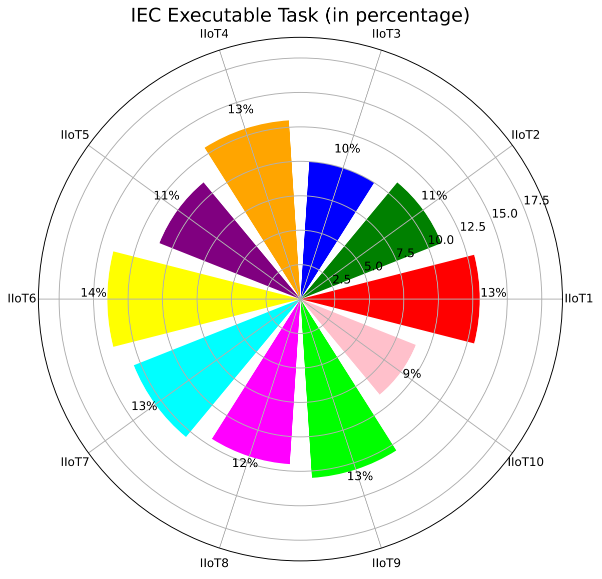


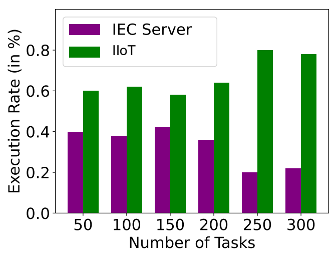
Fig. 2: Task classification strategy using `De-Sparse`.

servers, have been meticulously generated and carefully preserved. This ensures they are accurately labelled with their corresponding file names.

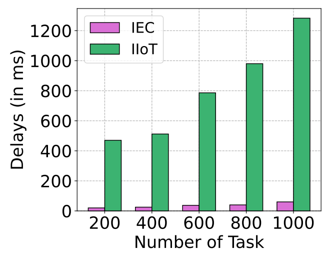
C. Performance Analysis

The simulation procedure was conducted to monitor and assess the performance of our proposed model compared to random offloading and existing standard methods. To facilitate this evaluation, we created a set of 100 distinct tasks for each IIoT system. Initially, using our novel `De-Sparse` binary offloading decision algorithm, we partitioned each task into executable components designated for the IEC system and the respective IIoT. The task classification performance and workload distribution among the IIoT devices are shown in Fig 2. Subsequently, the executable task for the IEC system was offloaded to the most suitable IEC server utilizing the FL global model.

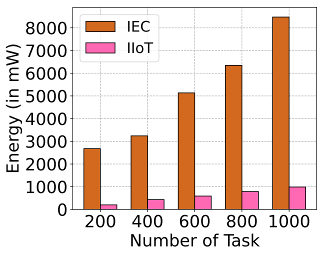
Fig. 3(a) shows the number of industrial tasks executed on IIoT devices and IEC servers. Fig. 3(b) and Fig. 3(c) show the delay performance and energy performance analysis of our proposed task offloading strategy. It can be observed that the overall delay on IIoT servers is comparatively high compared to IEC servers. However, opposite results have been observed in terms of overall energy consumption on various devices. The underlying cause lies in the extended transmission of data across vast distances between industrial devices and IEC servers, coupled with the demanding execution of resource-intensive operations within IEC servers. On the other hand, Fig. 3(d) illustrates a comparative analysis of our proposed strategy with existing standard algorithms. From Fig. 3(d), it is easy to validate that the overall system cost with our proposed customer strategy is minimal compared to standard baseline algorithms. Additionally, Fig. 4 illustrates the convergence performance of the `FedOff` task offloading method,



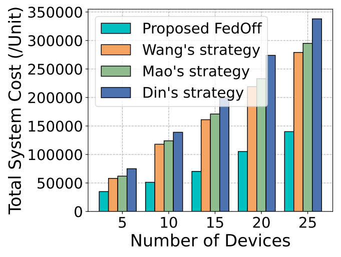
(a)



(b)



(c)



(d)

Fig. 3: Performance Analysis (a) Task completion rate, (b) Overall execution delay, (c) Energy consumption rate, (d) Total system cost and comparison with other methods.

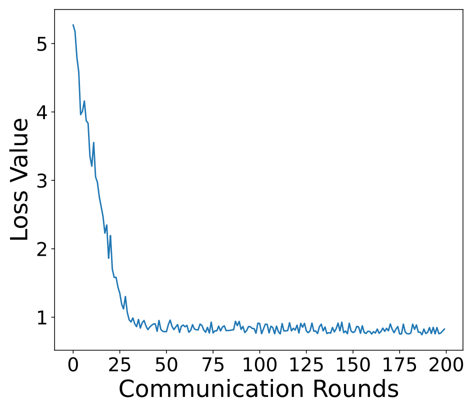


Fig. 4: Convergence of the `FedOff` task offloading method.

demonstrating improved decision-making capabilities over the communication rounds.

V. Conclusion

This paper addressed the issue of cost-optimized task offloading in industrial edge networks and proposed a novel `FedOff` task offloading framework to optimize task execution costs. Specifically, our proposed framework helps in identifying executable tasks on either IIoT devices or edge (*i.e.,* IEC), and consequently, offloaded tasks to suitable IEC while optimizing energy and delay. We formulated our cost optimization function by combining weighted energy and delay. In brief, we proposed the `De-Sparse` approach for classifying IIoT tasks into IIoT executable and IEC executable. Then, we utilized FL to offload IEC executable tasks to suitable devices while maintaining low network latency and energy consumption. A comprehensive simulation study demonstrated that the `FedOff` algorithm outperforms conventional algorithms in terms of energy and latency.

Acknowledgement

This work is supported by the National University of Singapore under Grant No.: MoE AcRF Tier-1 FRC Grant, NUS WBS No. A-0005142-01-00.

References

[1] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, "Joint Task Offloading and Resource Allocation for Delay-Sensitive Fog Networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[2] A. Hazra, M. Adhikari, S. Nandy, K. Doulani, and V. G. Menon, "Federated-Learning-Aided Next-Generation Edge Networks for Intelligent Services," *IEEE Network*, vol. 36, no. 3, pp. 56–64, 2022.

[3] A. Hazra, "Promising Role of Visual IoT: Challenges and Future Research Directions," *IEEE Engineering Management Review*, pp. 1–7, 2023.

[4] M. Mukherjee, V. Kumar, S. Kumar, C. X. Mavromoustakis, Q. Zhang, and M. Guo, "RIS-assisted Task Offloading for Wireless Dead Zone to Minimize Delay in Edge Computing," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 2554–2559.

[5] M. Guo, M. Mukherjee, Q. Guan, J. Ou, and C. Fan, *IEEE Transactions on Green Communications and Networking, title=Delay-Based Packet-Granular QoS Provisioning for Mixed Traffic in Industrial Internet of Things*, vol. 6, no. 4, pp. 2128–2143, 2022.

[6] A. Hazra and T. Amgoth, "CeCO: Cost-Efficient Computation Offloading of IoT Applications in Green Industrial Fog Networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6255–6263, 2022.

[7] M. Aazam, S. u. Islam, S. T. Lone, and A. Abbas, "Cloud of Things (CoT): Cloud-Fog-IoT Task Offloading for Sustainable Internet of Things," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 1, pp. 87–98, 2022.

[8] R. Kumar and N. Agrawal, "Rbac-lbrm: An rbac-based load balancing assisted efficient resource management framework for iot-edge-fog network," *IEEE Sensors Letters*, vol. 6, no. 8, pp. 1–4, 2022.

[9] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[10] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[11] Q. Wang, S. Guo, J. Liu, and Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing," *Sustainable Computing: Informatics and Systems*, vol. 21, pp. 154–164, 2019.

[12] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic Joint Radio and Computational Resource Management for Multi-User Mobile-Edge Computing Systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[13] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[14] J. Xia, P. Wang, B. Li, and Z. Fei, "Intelligent task offloading and collaborative computation in multi-uav-enabled mobile edge computing," *China Communications*, vol. 19, no. 4, pp. 244–256, 2022.