Cooperative Transmission Scheduling and Computation Offloading With Collaboration of Fog and Cloud for Industrial IoT Applications

Abhishek Hazra, *Student Member, IEEE*, Praveen Kumar Donta, *Member, IEEE*, Tarachand Amgoth, *Member, IEEE*, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*—Energy consumption for large amounts of delay-sensitive applications brings serious challenges with the continuous development and diversity of Industrial Internet of Things (IIoT) applications in fog networks. In addition, conventional cloud technology cannot adhere to the delay requirement of sensitive IIoT applications due to long-distance data travel. To address this bottleneck, we design a novel energy–delay optimization framework called transmission scheduling and computation offloading (`TSCO`), while maintaining energy and delay constraints in the fog environment. To achieve this objective, we first present a heuristic-based transmission scheduling strategy to transfer IIoT-generated tasks based on their importance. Moreover, we also introduce a graph-based task-offloading strategy using constrained-restricted mixed linear programming to handle high traffic in rush-hour scenarios. Extensive simulation results illustrate that the proposed `TSCO` approach significantly optimizes energy consumption and delay up to 12%–17% during computation and communication over the traditional baseline algorithms.**

***Index Terms*—Energy efficiency, fog computing, Industrial Internet of Things (IIoT), mixed linear programming, task offloading.**

I. Introduction

INDUSTRIAL Internet of Things (IIoT) has contributed toward the rapid growth in various industrial application domains, such as the green infrastructure, smart grid, smart city, smart transport networks, amongst others [1]. With these diverse applications, IIoT devices are also generating a massive amount of sensitive data, requiring immediate processing near edge devices, resulting in a shortage of lower storage and faster data processing among the IIoT devices. In such circumstances, transferring a portion of excessive data to a resource-rich remote computing device, also called *computation offloading*, is a suitable solution to handle sensitive IIoT applications [2]. In general, resource-constrained IIoT applications offload data to a specific cloud server for processing and data analysis [3]. However, several challenges are admitted due to the physical distance between the IIoT devices and the cloud servers. To overcome the shortcoming, CISCO (New York, 2012 [4]) introduced the fog-computing paradigm as an auxiliary tier to conventional cloud-computing technology to process delay sensitive, i.e., emergency IIoT applications in nearby edge devices [5]. Essentially, fog devices are used to deploy at the edge of the networks to optimize the overall latency and increase the reliability of the industrial network [6]. Thus, to gratify Quality-of-Service (QoS) objectives and to process emergency applications, a hierarchical fog–cloud environment is more beneficial for IIoT applications, where cloud servers can handle resource-hungry applications and fog devices can process other delay-sensitive applications simultaneously [7].

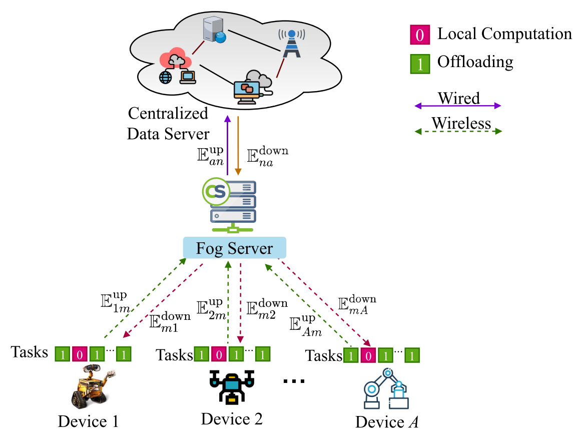


Fig. 1. Hierarchical IIoT–fog–cloud architecture.

A. Motivation

To demonstrate the motivation for our work, let us examine an example shown in Fig. 1. Let an industrial fog network consist of #1 fog device, #1 cloud server, and *A* number of IIoT devices, where each IIoT device generates five tasks from different sensors. Consider the uplink and downlink energy consumption between fog device and IIoT devices is a constant unit 1. For this example, complete uploading and

Manuscript received 12 November 2021; revised 30 January 2022; accepted 6 February 2022. Date of publication 9 February 2022; date of current version 20 February 2023. This work was supported by the Department of Science and Technology (DST) (Science and Engineering Research Board), Government of India under Grant EEQ/2018/000888. *(Corresponding author: Praveen Kumar Donta.)*

Abhishek Hazra and Tarachand Amgoth are with the Computer Science and Engineering Department, Indian Institute of Technology (Indian School of Mines) Dhanbad, Dhanbad 826004, India (e-mail: abhishekhazra.18DR0018@cse.iitism.ac.in; tarachand@iitism.ac.in).

Praveen Kumar Donta and Schahram Dustdar are with Distributed Systems Group, Vienna University of Technology, 1040 Vienna, Austria (e-mail: pdonta@dsg.tuwien.ac.at; dustdar@dsg.tuwien.ac.at).

Digital Object Identifier 10.1109/JIOT.2022.3150070

downloading energy consumption is considered the number of hopes between IIoT devices and computing devices. For quick comprehension, we admit equal numbers of computation-intensive and delay-sensitive tasks, and the processing energy required to execute these tasks is 1 and 2 units, respectively. Finally, we assume Fog devices can assign at most 2 units of computation resource to each IIoT device to perform the tasks, and the remaining tasks are uploaded to the cloud server.

In this example, we consider one IIoT device executes one task and offloads the remaining four tasks for remote execution. The energy dissipation for each IIoT device can now be calculated as $(1+2+2+1+1) = 7$. Given 2 available free units in fog devices, the task-allocator assigns delay-sensitive tasks to the fog device and computation-intensive tasks to the cloud server with energy consumption rate $(1 + 1) = 2$ and $(2+2) = 4$ units, respectively. Then, the total energy consumption to execute a delay-sensitive task to a fog device is $(1 + 1 + 1) = 3$ and computation-intensive task is $(1+1+2+1+1) = 6$. Thus, the total energy consumption to process all 5 tasks for an IIoT device is $(1+6+6+3+3) = 19$. If there are A numbers of IIoT devices, then the expected total energy consumption by the network is $(A \times 19)$. If we use a random task-offloading strategy and execute one computation-intensive task to the fog device, executing that task requires $(1 + 2 + 2) = 4$ units of energy, which is minimum. Still, overall energy consumption to execute all the tasks will be $(1+4+6+5+5) = 21$. For large IIoT devices, total energy consumption will become $(A\times 21)$, which is 11% more than the proposed solution. If the number of tasks in each IIoT device increases, the total energy consumption rate will increase by up to 30%–40%.

From this illustration, we can analyze that even a better strategy of fog association, transmission scheduling, and computation offloading not only minimizes the energy consumption rate but also increases the QoS for delay-critical IIoT applications.

B. Related Works

Over the last few years, numerous research efforts have been made to address issues related to computation offloading in the multitire fog–cloud architecture for handling various delay-restricted IIoT applications [8]. In this viewpoint, an apparent answer is to offload resource-hungry tasks to the cloud server or higher resource-oriented computing devices. For example, in [9], Hazra *et al.* have proposed an energy-optimized computation offloading strategy in stochastic fog networks. A code-oriented multiuser computation offloading approach has been designed by Ding *et al.* [10] for optimizing the execution overhead in mobile-edge computing (MEC) networks. Similarly, in [11], Mukherjee *et al.* have also introduced a deadline-aware computation offloading system for industrial fog networks. These works separately consider the delay and energy consumption rate for fog networks. However, they do not highlight the actual tradeoff between energy and latency. To address this issue, Sarkar *et al.* [3] have proposed a priority-based task scheduling and resource-based computation offloading strategy for delay-restricted IIoT applications. Sheng *et al.* [12] have also presented an energy-efficient partial computation offloading method in the collaboration of fog–cloud networks.

TABLE I
COMPARATIVE STUDY WITH THE EXISTING ALGORITHMS

Existing works	Transmission scheduling	Trade-off analysis	Complexity analysis	Computation offloading	Device selection
[3]	×	×	×	✓	✓
[9]	✓	×	✓	✓	×
[16]	✓	×	✓	✓	×
[18]	✓	×	×	✓	×
[19]	×	×	✓	✓	✓
[20]	×	×	×	✓	✓
[22]	×	×	×	✓	×
[23]	×	✓	✓	×	×
Our work	✓	✓	✓	✓	✓

Offloading application data from IIoT devices to a remote computing server can certainly decrease execution time and overcome energy usage in the industrial environment [13]. However, remote execution is not always a viable option because remote server processing necessitates additional data transmission delay, which might lengthen the overall execution duration and drain the battery of the IIoT devices [14]. To address device selection challenges and catch advantage of the offloading mechanism, Yadav *et al.* [15] have outlined a latency-driven task placement strategy for IoT applications. A graph-based computation offloading strategy has been introduced by Sarkar *et al.* [3] for optimizing the computation overhead over the federated fog networks. Similarly, Hazra *et al.* [16] and Li *et al.* [17] have introduced several optimization techniques and offloading mechanisms to minimize the energy–delay for the execution of IIoT applications. A summary of the existing contributions is presented in Table I.

Most of the current strategies focus on optimal scheduling and computation offloading approaches separately for reaching various QoS objectives, including minimizing delay and energy consumption [1], [18]. Nonetheless, the earlier studies do not consider the importance of transmission scheduling and the device-matching strategy in the industrial fog networks, even though the optimal device-matching strategy helps to offload tasks in suitable computing devices and utilize fog resources more efficiently [19]. On the other hand, transmission scheduling helps control priority-driven data transmission over fog networks. Therefore, all these challenges encourage to design of cooperative transmission scheduling and computation offloading (TSCO) strategies for industrial applications, where emergency tasks can be prioritized depending on network conditions. Hence, there are two significant challenges for offloading computation data through a hierarchical fog network. First, how to determine an efficient transmission scheduling strategy for delay-sensitive IIoT applications so that the system can find an efficient scheduling order pair for all IIoT-generated tasks. Second, how to define an optimal task-device-matching strategy in the fog networks so that IIoT-generated tasks are adequately offloaded to suitable devices.

C. Contributions

Considering these challenges in mind, we propose an efficient TSCO scheme for minimizing the overall delay and energy consumption rate of industrial fog networks. Specifically, the notable contributions of this article are listed as follows.

1) This article aims at designing a "green" IIoT system called TSCO for handling various emergency tasks in the fog environment. First, we define a transmission scheduling policy for all IIoT-generated tasks based on their input data size and transmission rate of industrial devices. This strategy also considers the current network dynamics to reduce the transmission overhead through the network.
2) To maximize the utilization of fog/cloud resources, we also introduce a device adaptation strategy called the device-matching order (DMO). Our proposed graph-based decision-making system is also handy in making near-optimal offloading decisions among the computing devices. A theoretical analysis is also performed to determine the energy–delay tradeoff of the proposed model.
3) Extensive simulation and performance analysis demonstrate that our proposed TSCO strategy is suitable to overcome the average waiting time, processing time, and energy consumption rate over the existing baseline algorithms.

The remainder of this article is structured as follows. Section II discusses the mathematical modeling of fog networks. The energy-aware computation offloading strategy is exhibited in Section III. The numerical analysis of our proposed TSCO approach is explained in Section IV. Finally, conclusions and future research objectives are considered in Section V.

II. Computation Offloading Model

Considering an industrial fog–cloud network with a finite number of M fog devices, denoted as $\mathcal{M} = \{1, 2, \dots, M\}$ $\forall m \in \mathcal{M}$ and each m contains multiple processing instances. In this network, let A denote the set of IIoT devices, represented as $\mathcal{A} = \{1, 2, \dots, A\}$ $\forall a \in \mathcal{A}$ and N be the set of cloud servers, denoted as $\mathcal{N} = \{1, 2, \dots, N\}$ $\forall n \in \mathcal{N}$. Each IIoT device can generate K, $\mathcal{K} = \{1, 2, \dots, K\}$ $\forall k \in \mathcal{K}$ number of time-dependent tasks with input and output data size $\mathcal{K}_k^{\text{in}}$ and $\mathcal{K}_k^{\text{out}}$ (in bits), respectively. The tasks can process locally or offload to the nearby fog device/cloud server for further processing through G numbers of gateway devices, represented as $\mathcal{G} = \{1, 2, \dots, G\}$ $\forall g \in \mathcal{G}$. Here, we consider a binary offloading situation, where IIoT devices deploy entire tasks to the available computing devices $\mathcal{S}_l$ $\forall l \in (\mathcal{M} \cup \mathcal{N})$ based on multiple QoS parameters. Denote $\mathcal{X} \in \mathbb{R}^{\mathcal{K} \times (\mathcal{A} \cup \mathcal{M} \cup \mathcal{N})}$ as a task allocation matrix, where the (k, l)th entry is defined by

$$\mathcal{X}(k, l) = \begin{cases} 1, & \text{if } k\text{th task is assigned to } l\text{th device} \\ 0, & \text{otherwise.} \end{cases}$$

We consider a task k $\forall k \in \mathcal{K}$ contains three *attributes* while generating, i.e., $K_k^{\text{in}} = \langle K_k^{\text{CPU}}, K_k^{\text{freq}}, K_k^{\text{exe}} \rangle$, where K_k^{CPU} denotes the CPU requirement, and K_k^{freq} represents the variable-length task generating frequency, whereas K_k^{exe} represents the execution deadline of task k $\forall k \in \mathcal{K}$. Furthermore, we consider that gateway devices $\mathcal{G}$ request services to multiple fog devices $\mathcal{M}$, consequently each fog device m $\forall m \in \mathcal{M}$ also receives multiple requests from IIoT devices $\mathcal{A}$ at time t $\forall t \in \mathcal{T}$, where $\mathcal{T} = \{1, 2, \dots, T\}$. The important notations are referred to Table II.

TABLE II
Frequently Used Notations

Symbols	Definition
$\mathcal{K}$	Set of IIoT tasks in the fog networks
$\mathcal{N}$	Set of cloud servers in the fog networks
$\mathcal{M}$	Set of fog devices in the industrial networks
$\mathcal{A}$	Set of IIoT devices in the fog networks
$\mathcal{G}$	Set of IIoT gateway devices in the fog networks
λ_k	Task arrival rate in the IIoT device $\mathcal{A}$
$\mathcal{S}_l$	Set of processing devices in the network
$\mathscr{R}_{am}^{\text{up}}$	Data transmission rate from ath IIoT to mth fog device
$\mathcal{X}(k, l)$	Binary computation offloading decision matrix
$\mathscr{H}_a$	Channel power gain of the IIoT device
Γ_m^{CPU}	Computation frequency of the fog device
$\mathscr{B}_{am}^{\text{up}}$	Transmission bandwidth between IIoT device and fog device
$\mathbb{E}_{ka}^{\text{process}}$	Energy consumption on local IIoT device
$\mathcal{E}_l^{max}$	Maximum energy consumption threshold
$\mathcal{T}_l^{max}$	Maximum tolerable delay on computing device l
$\mathbb{E}_{kl}^{\text{total}}$	Total energy consumption on computing device l
$\mathscr{P}_a^{\text{process}}$	Predefined energy consumption rate for IIoT devices $\mathcal{A}$

A. Local Execution

First, IIoT devices $\mathcal{A}$ check the availability of the CPU frequency on their own. If the CPU frequency of the ath IIoT device Γ_a^{CPU} satisfy the tasks CPU requirement K_k^{CPU}, i.e., $\Gamma_a^{\text{CPU}} > K_k^{\text{CPU}}$, then IIoT devices execute tasks locally. Let φ be the processing density for the kth task. Thus, the task processing time $\mathbb{T}_{ka}^{\text{process}}$ on the local IIoT device $a \in \mathcal{A}$ can be expressed as follows:

$$\mathbb{T}_{ka}^{\text{process}} = \frac{\mathcal{X}(k, a) \times \varphi . K_k^{\text{in}}}{\Gamma_a^{\text{CPU}}}. \tag{1}$$

Similarly, the energy consumption $\mathbb{E}_{ka}^{\text{process}}$ to process a task $k \in \mathcal{K}$ in the IIoT device $a \in \mathcal{A}$ is given as follows:

$$\mathbb{E}_{ka}^{\text{process}} = \mathbb{T}_{ka}^{\text{process}} \times \mathscr{P}_a^{\text{process}} \tag{2}$$

where $\mathscr{P}_a^{\text{process}}$ defines the predefined energy consumption rate for IIoT devices $\mathcal{A}$ deployed in the industrial networks.

B. Fog Execution

Recent advancements in storage technology allow IIoT devices to process a small portion of tasks locally. However, due to the limited CPU frequency of IIoT devices, tasks are forwarded to suitable computing devices that should satisfy minimum latency and available resource requirements. Denote $\mathscr{H}_a$ and $\mathscr{P}_a^{\text{up}}$ as the channel power gain and transmission power of the ath IIoT device. By considering the Shannon capacity formula [23], the uploading data transmission rate $\mathscr{R}_{am}^{\text{up}}$ is defined as $\mathscr{R}_{am}^{\text{up}} = \mathscr{B}_{am}^{\text{up}} \log_2(1 + [\mathscr{P}_a^{\text{up}} \mathscr{H}_a / \xi_m^2])$, where $\mathscr{B}_{am}^{\text{up}}$ signifies the allocated transmission bandwidth between the ath IIoT device and the mth fog device. Thus, the data transmission time $\mathbb{T}_{am}^{\text{up}}$ and transmission energy usage $\mathbb{E}_{am}^{\text{up}}$ to a fog device $m \in \mathcal{M}$ can be represented as follows:

$$\mathbb{T}_{am}^{\text{up}} = \frac{\mathcal{X}(k,m) \times \varphi.K_k^{\text{in}}}{\mathscr{R}_{am}^{\text{up}}} \tag{3}$$

$$\mathbb{E}_{am}^{\text{up}} = \mathbb{T}_{am}^{\text{up}} \times \mathscr{P}_a^{\text{up}}. \tag{4}$$

Once tasks are received, fog devices $\mathcal{M}$ immediately start their execution process. Thus, the total processing delay $\mathbb{T}_{am}^{\text{process}}$, i.e., time taken to process the kth task in the mth fog device is given as follows:

$$\mathbb{T}_{am}^{\text{process}} = \frac{\mathcal{X}(k,m) \times \varphi.K_k^{\text{in}}}{\Gamma_m^{\text{CPU}}} \tag{5}$$

where Γ_m^{CPU} represents the computational capacity of the mth fog device. From (5), we can derive the overall energy consumption rate on the fog device $m \in \mathcal{M}$ as follows:

$$\mathbb{E}_{am}^{\text{process}} = \frac{\mathcal{X}(k,m) \times \varphi.K_k^{\text{in}}}{\Gamma_m^{\text{CPU}}} \times \mathbb{E}_m^{\text{CPU}}. \tag{6}$$

Similarly, let $\mathscr{R}_{ma}^{\text{down}} = \mathscr{B}_{ma}^{\text{down}} \log_2(1 + [\mathscr{P}_m^{\text{down}}\mathscr{H}_m/\xi_a^2])$ be the down transmission rate to the IIoT device, where $\mathscr{B}_{ma}^{\text{down}}$ denotes the available transmission bandwidth utilization between the mth fog device to the ath end device. Then, the downloading time $\mathbb{T}_{ma}^{\text{down}}$ and energy usage $\mathbb{E}_{ma}^{\text{down}}$ to fetch the results at the IIoT device $m \in \mathcal{M}$ can be expressed as follows:

$$\mathbb{T}_{ma}^{\text{down}} = \frac{\mathcal{X}(m,k) \times K_k^{\text{out}}}{\mathscr{R}_{ma}^{\text{down}}} \tag{7}$$

$$\mathbb{E}_{ma}^{\text{down}} = \mathbb{T}_{ma}^{\text{down}} \times \mathscr{P}_m^{\text{down}} \tag{8}$$

where $K_k^{\text{out}} = \Upsilon \times K_k^{\text{in}}$ and Υ is the scaling coefficient. Thus, the total delay and energy consumed by the task $k \in \mathcal{K}$ while processing in a fog device $m \in \mathcal{M}$ is defined as $\mathbb{T}_{am}^{\text{total}} = \mathbb{T}_{am}^{\text{up}} + \mathbb{T}_{am}^{\text{process}} + \mathbb{T}_{ma}^{\text{down}}$ and $\mathbb{E}_{am}^{\text{total}} = \mathbb{E}_{am}^{\text{up}} + \mathbb{E}_{am}^{\text{process}} + \mathbb{E}_{ma}^{\text{down}}$.

C. Cloud Execution

Let $\mathscr{H}_a$ and $\mathscr{P}_a^{\text{up}}$ be the channel power gain and transmission power of the ath IIoT device. Then, the uploading data transmission rate $\mathscr{R}_{an}^{\text{up}}$ to a cloud server $n \in \mathcal{N}$ is defined as $\mathscr{R}_{an}^{\text{up}} = \mathscr{B}_{an}^{\text{up}} \log_2(1 + [\mathscr{P}_a^{\text{up}}\mathscr{H}_a/\xi_n^2])$, where $\mathscr{B}_{an}^{\text{up}}$ defines the transmission bandwidth between the ath IIoT device and the nth cloud server. Thus, the transmission delay $\mathbb{T}_{an}^{\text{up}}$ and energy consumption $\mathbb{E}_{an}^{\text{up}}$ on a cloud server $n \in \mathcal{N}$ can be defined as follows:

$$\mathbb{T}_{an}^{\text{up}} = \frac{\mathcal{X}(k,n) \times \varphi.K_k^{\text{in}}}{\mathscr{R}_{an}^{\text{up}}} \tag{9}$$

$$\mathbb{E}_{an}^{\text{up}} = \mathbb{T}_{an}^{\text{up}} \times \mathscr{P}_a^{\text{up}}. \tag{10}$$

Consequently, the total time and energy are taken to process the kth task in the nth cloud server is expressed as follows:

$$\mathbb{T}_{an}^{\text{process}} = \frac{\mathcal{X}(k,n) \times \varphi.K_k^{\text{in}}}{\Gamma_n^{\text{CPU}}} \tag{11}$$

$$\mathbb{E}_{an}^{\text{process}} = \frac{\mathcal{X}(k,n) \times \varphi.K_k^{\text{in}}}{\Gamma_n^{\text{CPU}}} \times \mathbb{E}_n^{\text{CPU}} \tag{12}$$

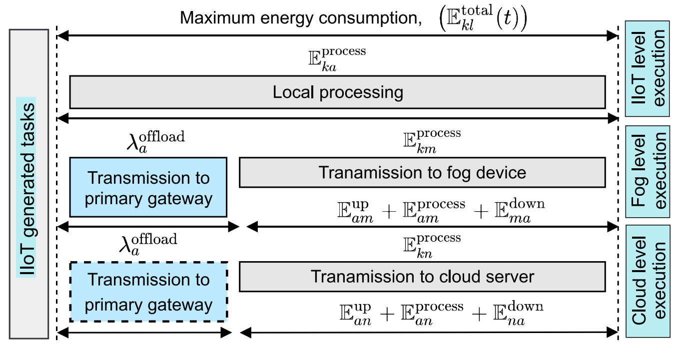


Fig. 2. Illustration of task-offloading decision.

where $\mathbb{E}_n^{\text{CPU}}$ be the energy consumption rate at the cloud server. The achievable downloading rate $\mathscr{R}_{na}^{\text{down}}$ to the IIoT device $a \in \mathcal{A}$ can be expressed as $\mathscr{R}_{na}^{\text{down}} = \mathscr{B}_{na}^{\text{out}} \log_2(1 + [\mathscr{P}_n^{\text{down}}\mathscr{H}_n/\xi_a^2])$. Similarly, the downloading time $\mathbb{T}_{na}^{\text{down}}$ and energy consumption $\mathbb{E}_{na}^{\text{down}}$ of task $k \in \mathcal{K}$ can be expressed as follows:

$$\mathbb{T}_{na}^{\text{down}} = \frac{\mathcal{X}(n,k) \times K_k^{\text{out}}}{\mathscr{R}_{na}^{\text{down}}} \tag{13}$$

$$\mathbb{E}_{na}^{\text{down}} = \mathbb{T}_{na}^{\text{down}} \times \mathscr{P}_n^{\text{down}}. \tag{14}$$

Thus, the overall delay and energy usage on a cloud server $n \in \mathcal{N}$ is defined as $\mathbb{T}_{an}^{\text{total}} = \mathbb{T}_{an}^{\text{up}} + \mathbb{T}_{an}^{\text{process}} + \mathbb{T}_{na}^{\text{down}}$ and $\mathbb{E}_{an}^{\text{total}} = \mathbb{E}_{an}^{\text{up}} + \mathbb{E}_{an}^{\text{process}} + \mathbb{E}_{na}^{\text{down}}$. From the above formulations, we can derive the overall delay and energy consumption to process a task $k \in \mathcal{K}$ on IIoT devices $\mathcal{A}$ and other computing devices $\mathcal{S}_l$ $\forall l \in (\mathcal{M} \cup \mathcal{N})$ can be expressed as follows:

$$\mathbb{T}_{kl}^{\text{total}} = \begin{cases} \mathbb{T}_{al}^{\text{process}}, & \text{if } l \in \mathcal{A} \\ \mathbb{T}_{al}^{\text{up}} + \mathbb{T}_{al}^{\text{process}} + \mathbb{T}_{la}^{\text{down}}, & \text{if } l \in \mathcal{S}_l \end{cases} \tag{15}$$

$$\mathbb{E}_{kl}^{\text{total}} = \begin{cases} \mathbb{E}_{al}^{\text{process}}, & \text{if } l \in \mathcal{A} \\ \mathbb{E}_{al}^{\text{up}} + \mathbb{E}_{al}^{\text{process}} + \mathbb{E}_{la}^{\text{down}}, & \text{if } l \in \mathcal{S}_l. \end{cases} \tag{16}$$

D. Problem Formulation

The primary objective following this problem formulation is to find a near-optimal scheduling order minimizing the proposed objective function while considering energy as the primary concern, as illustrated in Fig. 2. The objective function has two perspectives, i.e., minimize energy consumption rate and reduce overall processing time. The above goals and correlated constraints are theoretically formulated as follows:

$$\text{minimize} \quad \lim_{t\to\infty} \sum_{t\in T} \alpha \cdot \mathbb{E}_{kl}^{\text{total}}(t) + \beta \cdot \mathbb{T}_{kl}^{\text{total}}(t) \tag{17a}$$

$$\text{subject to} \quad 0 \le \mathbb{E}_{kl}^{\text{total}}(t) \le \mathcal{E}_l^{\max} \tag{17b}$$

$$0 \le \mathbb{T}_{kl}^{\max}(t) \le \mathcal{T}_l^{\max} \tag{17c}$$

$$0 \le \Gamma_k^{\text{CPU}}(t) \le \Gamma_l^{\max} \tag{17d}$$

$$\sum_{k\in|\mathcal{K}|} \sum_{l\in|\mathcal{S}_l|} \mathcal{X}(k,l) \le |\mathcal{S}_l| \tag{17e}$$

$$\sum_{k\in|\mathcal{K}|} \mathcal{X}(k,l) = 1 \tag{17f}$$

$$\mathcal{X}(k,l) \in \{0,1\} \tag{17g}$$

$$\mathbb{T}_{kl}^{\text{up}} \ge 0 \text{ and } \mathbb{T}_{lk}^{\text{down}} \ge 0 \tag{17h}$$

where $\alpha + \beta = 1$. Constraint (17b) states the overall energy consumption of the kth task is less than the maximum energy consumption $\mathcal{E}_l^{\max}$ of the lth computing device. Constraint (17c) restricts the total processing delay to a maximum tolerable delay $\mathcal{T}_l^{\max}$ on the lth device. Constraint (17d) clarifies that the maximum processing frequency of task $k \in \mathcal{K}$ is less than the maximum tolerable frequency $\Gamma_l^{\max}$ on device $l\ \forall l \in (\mathcal{A} \cup \mathcal{M} \cup \mathcal{N})$. Constraint (17e) defined that each task should be allocated at-most one computing device $l \in \mathcal{S}_l$ at time t. Constraint (17f) restricts the task-offloading value to maximum 1 and constraint (17g) imposes the binary offloading constraint. Finally, constraint (17h) signifies a non *zero* data transmission time among the computing devices.

III. Energy Efficient Offloading Strategy

In this section, we aim to collectively optimize the distribution of communication and computation resources in virtual computing devices (both fog devices and cloud servers) to achieve the least possible delay and energy consumption over the fog networks. To confirm this, we divide our computation offloading strategy into two phases. In the first phase, a network-dependent transmission scheduling scheme is introduced. Then, to offload the scheduled tasks, a mixed-integer programming scheme is adopted to allocate tasks on suitable computing devices, discussed as follows.

A. Index-Based Transmission Scheduling

Recognizing the index of each IIoT device $a \in \mathcal{A}$ and accordingly allow them for transmission is the preprocessing step of our proposed transmission scheduling technique (IBTS). Without loss of generality, we consider that the system follows a static-index-based ranking policy. Initially, IIoT devices $\mathcal{A}$ store all the generated tasks $\mathcal{K}$ in a local queue. Let λ_k be the dynamic task arrival rate at any IIoT device a $\forall a \in \mathcal{A}$. Denote $1/\beta_a = \mathbb{T}_{al}^{\text{up}}$ as the average upstream transmission time and $\lambda_k/\beta_a = (\mathcal{X}(k,l) \times \lambda_k)/\beta_a$ as the traffic intensity of the ath IIoT device defined in [24]. Furthermore, let $\mathcal{K}_k^{\text{in}}(t)$ be the amount of task buffered in the ath IIoT device at a time instance t. Thus, we have a delay-dependent priority indexing $\mathscr{C}_a^{\text{index}}(t)$ for each task $k\ \forall k \in \mathcal{K}$ at the initial stages of time t, which can be expressed as follows:

$$\begin{aligned}\mathscr{A} &= \arg\max_{a\in\mathcal{A}} \mathscr{C}_a^{\text{index}}(t)\\ &= \arg\max_{a\in\mathcal{A}} \left(\frac{\mathcal{K}_k^{\text{in}}(t)}{\mathcal{X}(k,a)\times\lambda_k}\right)\beta_a. \end{aligned} \tag{18}$$

Definition: A task k generated through IIoT device $a \in \mathcal{A}$ is called delay sensitive task $\mathcal{K}_k^D$, if the device priority index $\mathscr{C}_a^{\text{index}}$ is less than or equal to $\mathscr{D}$, i.e., $\mathscr{C}_a^{\text{index}} \le \mathscr{D}$. Otherwise, the task is classified as resource-intensive task $\mathcal{K}_k^R$.

1) *Illustration Example:* Considering two IIoT devices IIoT#1 and IIoT#2 are actively generating data with transmission time $1/\beta_1 = 0.2$ s and $1/\beta_2 = 0.4$ s, respectively. Assuming that task-offloading rate of IIoT#1 = 3 tasks/s and IIoT#2 = 2 tasks/s, respectively, with equal probability. If at time t, the number of tasks stored in a local queue of IIoT gateway is $\mathcal{K}_1^{\text{in}}(t) = 5$ and $\mathcal{K}_2^{\text{in}}(t) = 3$. Then, according to (18), we have the following priority index of IIoT#1 and IIoT#2 with indexing threshold $\mathscr{D} = 0.5$ as follows:

$$\mathscr{C}_1^{\text{index}}(t) = (5/3) \times 0.2 = 0.33 \tag{19}$$

$$\mathscr{C}_2^{\text{index}}(t) = (3/2) \times 0.4 = 0.60. \tag{20}$$

Since $\mathscr{C}_1^{\text{index}}(t) \le 0.5$ and $\mathscr{C}_2^{\text{index}}(t) > 0.5$, IIoT#1 will be considered as delay sensitive and IIoT#2 will be considered as resource intensive starting from time t. In the following sections, we prove that this index-based transmission scheduling strategy asymptotically diminishes the overall execution delay of IIoT applications.

B. Device-Matching Order

This section introduces our proposed (DMO) policy for distributing all the scheduled tasks among suitable fog devices or cloud servers. Initially, we construct a $(k \times l)$ task assignment matrix $E_{k\times l}$, where the row indicates the tasks and column indicates the resources. Each entry in matrix E is denoted using the e_{kl} value, a nonnegative heuristic information for task k $(k \le K)$ to assign the resource l $(l \le \mathcal{S}_l$, where $\mathcal{S}_l = \mathcal{M} \cup \mathcal{N})$. Each entry e_{kl} in matrix $E_{k\times l}$ is computed using the following:

$$\min_{k\in\mathcal{K}} \ \alpha \cdot \mathbb{E}_{kl}^{\text{total}}(t) + \beta \cdot \mathbb{T}_{kl}^{\text{total}}(t). \tag{21}$$

Now, we can generate task allocation matrix $\mathcal{X}^*$ using assignment matrix E, where each entry of $\mathcal{X}^*$ is either 0 or 1. The goal of the DMO algorithm is shown in (22a)

$$\text{minimize} \ \max_{k\in\mathcal{K}} \sum_{l=1}^{\mathcal{S}_l} \mathcal{X}(k,l) \cdot e_{kl} \tag{22a}$$

$$\text{subjectto} \ \mathcal{X}(k,l) \in \{0,1\} \quad \forall l \in \mathcal{S}_l \tag{22b}$$

$$\sum_{l=1}^{\mathcal{S}_l} \mathcal{X}(k,l) = 1 \quad \forall k = \{1,2,\ldots,K\}. \tag{22c}$$

It is important to note that the DMO strategy tradeoff energy and delay for obtaining suitable computing devices from the device pool. The DMO policy performs the following steps in order to achieve the goal in (22a).

Step 1: Order the e_{kl} values in nondecreasing order, i.e., $e_{kl(1)} \le e_{kl(2)} \le e_{kl(3)} \le \cdots \le e_{kl(\mathcal{S}_l)}$.

Step 2: Find the minimum e_{kl} rank as an element (r) in the kth row and lth column E in increasing order until each column and row contains at least one element.

Step 3: Replace the entries e_{kl} of E according to

$$e_{kl} = \begin{cases} 0, & \text{if } e_{kl} \le r \\ e_{kl}, & \text{Otherwise.} \end{cases} \tag{23}$$

Step 4: Consider a column (l) which had less number of *zeros*, and assign all the tasks (k) whose associated values are 0 to the particular resource (either fog/cloud).

Step 5: Repeat step 4 until all the tasks are offloaded.

We illustrate the proposed DMO strategy through an example in Fig. 3 for better understanding. This example considers seven tasks generated by IIoT devices, three fog nodes, and two cloud resources. We assume each fog/cloud had multiple computing instances to process multiple tasks at a time. The

	F1	F2	F3	C1	C2
T1	25	65	32	65	72
T2	54	77	21	41	56
T3	75	55	58	87	36
T4	62	45	54	65	74
T5	15	44	34	69	85
T6	66	51	59	36	64
T7	28	57	33	57	81

(a)

	F1	F2	F3	C1	C2
T1	3	24	5	25	29
T2	14	32	2	10	17
T3	31	16	20	35	9
T4	22	12	15	26	30
T5	1	11	7	28	34
T6	27	13	21	8	23
T7	4	18	6	19	33

(b)

	F1	F2	F3	C1	C2
T1	3	–	5	–	–
T2	–	–	2	10	–
T3	–	–	–	–	9
T4	–	12	–	–	–
T5	1	11	7	–	–
T6	–	–	–	8	–
T7	4	–	6	–	–

(c)

	F1	F2	F3	C1	C2
T1	0	65	0	65	72
T2	54	77	0	0	56
T3	75	55	58	87	0
T4	62	0	54	65	74
T5	0	0	0	69	85
T6	66	51	59	0	64
T7	0	57	0	57	81

(d)

	F1	F2	F3	C1	**C2**
T1	0	65	0	65	72
T2	54	77	0	0	56
~~T3~~	~~75~~	~~55~~	~~58~~	~~87~~	~~**0**~~
T4	62	0	54	65	74
T5	0	0	0	69	85
T6	66	51	59	0	64
T7	0	57	0	57	81

(e)

	F1	**F2**	F3	C1	C2
T1	0	65	0	65	72
T2	54	77	0	0	56
~~T3~~	~~75~~	~~55~~	~~58~~	~~87~~	~~**0**~~
~~T4~~	~~62~~	~~**0**~~	~~54~~	~~65~~	~~74~~
~~T5~~	~~0~~	~~**0**~~	~~0~~	~~69~~	~~85~~
T6	66	51	59	0	64
T7	0	57	0	57	81

(f)

	F1	F2	F3	C1	C2
~~T1~~	~~**0**~~	~~65~~	~~0~~	~~65~~	~~72~~
T2	54	77	0	0	56
~~T3~~	~~75~~	~~55~~	~~58~~	~~87~~	~~**0**~~
~~T4~~	~~62~~	~~**0**~~	~~54~~	~~65~~	~~74~~
~~T5~~	~~0~~	~~**0**~~	~~0~~	~~69~~	~~85~~
T6	66	51	59	0	64
~~T7~~	~~**0**~~	~~57~~	~~0~~	~~57~~	~~81~~

(g)

	F1	F2	**F3**	C1	C2
~~T1~~	~~**0**~~	~~65~~	~~0~~	~~65~~	~~72~~
~~T2~~	~~54~~	~~77~~	~~**0**~~	~~0~~	~~56~~
~~T3~~	~~75~~	~~55~~	~~58~~	~~87~~	~~**0**~~
~~T4~~	~~62~~	~~**0**~~	~~54~~	~~65~~	~~74~~
~~T5~~	~~0~~	~~**0**~~	~~0~~	~~69~~	~~85~~
T6	66	51	59	0	64
~~T7~~	~~**0**~~	~~57~~	~~0~~	~~57~~	~~81~~

(h)

	F1	F2	F3	**C1**	C2
~~T1~~	~~**0**~~	~~65~~	~~0~~	~~65~~	~~72~~
~~T2~~	~~54~~	~~77~~	~~**0**~~	~~0~~	~~56~~
~~T3~~	~~75~~	~~55~~	~~58~~	~~87~~	~~**0**~~
~~T4~~	~~62~~	~~**0**~~	~~54~~	~~65~~	~~74~~
~~T5~~	~~0~~	~~**0**~~	~~0~~	~~69~~	~~85~~
~~T6~~	~~66~~	~~51~~	~~59~~	~~**0**~~	~~64~~
~~T7~~	~~**0**~~	~~57~~	~~0~~	~~57~~	~~81~~

(i)

	F1	F2	F3	C1	C2
T1	**1**	0	0	0	0
T2	0	0	**1**	0	0
T3	0	0	0	0	**1**
T4	0	**1**	0	0	0
T5	0	**1**	0	0	0
T6	0	0	0	**1**	0
T7	**1**	0	0	0	0

(j)

Fig. 3. Illustration of the DMO strategy through an example. (a) Initial value of matrix E, extracted using (21). (b) Ranks of each entry of the matrix E. (c) Fill the ranks in ascending order until each column and row contains at least an element. (d) Replace the ranks with zeros and reaming entries with the values similar to matrix E. (e) Consider column $C2$ because of least number of zeros. (f) Consider column $F2$. (g) Consider column $F1$. (h) Consider column $F3$. (i) Consider column $C1$. (j) Replace the assignment with ones and remaining with zeros.

rounded values get from (21) are considered as a matrix E as shown in Fig. 3(a). Now, we identify the ranks of each entry of the matrix E as shown in Fig. 3(b). Once the rank of each entry is identified, we fill one by one entry of the rank in nondecreasing order until each row and column can fill with at least an entry. It is deprecated in Fig. 3(c) and this condition satisfied once the rank 12 is filled. From Fig. 3(d), we can observe that the elements which are less than or equal to the value associated with rank 12, i.e., 45, are replaced with 0. Now we start assigning each task to a resource according to the *zeros* in the matrix by giving high priority to the least number of *zeros* in a column. So, column $C2$ contains the least number of 0, so the *zero*-associated task $T3$ is assigned to $C2$ and strike off the row (strike-off means it will not be considered during further assignments) as shown in Fig. 3(e). From Fig. 3(f), we note that the columns $C1$ and $F2$ contain the least number of *zeros*. Here, we can consider any one, but

Algorithm 1: TSCO Algorithm

INPUT: $\mathcal{K}_k^{in}$, λ_k, r, β_a, $\mathcal{X}(k,a)$, E_{kl}, $\mathscr{D}$
OUTPUT: Task offloading decision

```
Initialize K_k^in, λ_k, r, E_kl and D
Calculate C_a^index(t) using Eq.(18)
for a = 1 to A do
    Identify K_k^D and K_k^R using threshold D
    Offload K_k^R to cloud server n ∈ N
    for l = 1 to S_l do
        Sort(e_kl) // e_kl(1) ≤ e_kl(2) ≤ e_kl(3) ≤ ··· ≤ e_kl(S_l)
        Identify arg min{e_kl}
        Apply Eq. (23) to update e_kl, k ∈ K and l ∈ S_l
        while min{count_Zeros(l)}&k ≠ null do
            Assign task k ∈ K to resource l ∈ S_l
            free(k, l) in each step
        end while
        A ← A\{a} and S_l ← S_l\{l}
        Offload tasks to suitable computing devices
    end for
end for
```

giving the high priority to Fog nodes. So, the zero associated in the column $F2$, i.e., tasks $T4$ and $T5$ are assigned to it. Similarly, tasks $T1$ and $T7$ are assigned to $F1$ as shown in Fig. 3(g), $T2$ is assigned to $C1$ as shown in Fig. 3(h), and $T6$ is assigned to $C1$ as shown in Fig. 3(i). Furthermore, the task allocation matrix $\mathcal{X}^*$ is generated from the above offloading decisions as shown in Fig. 3(j). Detailed steps of the TSCO strategy are shown in Algorithm 1.

2) *Handling of Task-Offloading Failure Scenario:* Task failure is a critical issue in handling sensitive IIoT applications where each data contain some sensing or actuating information and need to be processed in the stipulated period. To address such circumstances, our proposed TSCO strategy first ranks IIoT devices based on the importance level of their data. Then, the DMO strategy is introduced to saturate k number of industrial tasks to $\mathcal{S}_l$ number of computing devices. Specifically, the DMO strategy transforms the task assignment problem into a graph-based problem, making the network simple to understand and offloading the task to suitable computing devices. On the other hand, unsuccessful tasks will wait for the subsequent iterations, allowing the network to track failure scenarios to some extent.

Theorem 1: Given a set of tasks $\mathcal{K}$ and active computing devices $\mathcal{S}_l$ $\forall l \in \{\mathcal{M} \cup \mathcal{N}\}$, the upper bound of task offloaded using the DMO strategy is $\min\{\Theta(|\mathcal{K}|), \Theta(|\mathcal{S}_l|)\}$.

Proof: The performance bound for a set of $|\mathcal{K}|$ scheduled nonpreemptive tasks on $|\mathcal{S}_l|$ $\forall l = \{\mathcal{M} \cup \mathcal{N}\}$, active computing devices, where $|.|$ denotes the cardinality of a set, can be determined by considering the three cases of the upper bound.

1) When $|\mathcal{K}| < |\mathcal{S}_l|$, the maximum number of tasks are upper bounded by $\Theta(|\mathcal{K}|)$, where $1 \leq |\mathcal{K}| \leq |\mathcal{S}_l|$ and only $|\mathcal{K}|$ tasks can be offloaded by the DMO strategy.

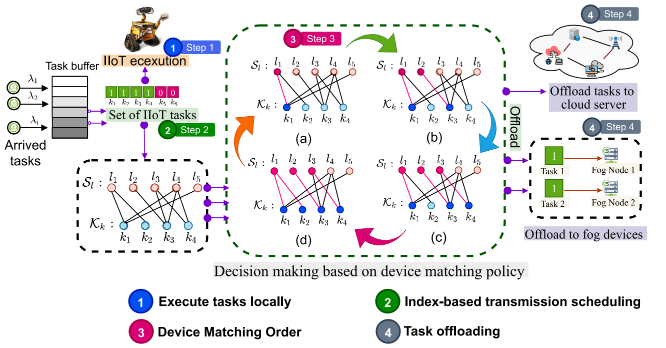


Fig. 4. Illustration of the proposed computation offloading strategy.

2) When $|\mathcal{K}| = |\mathcal{S}_l|$, the DMO strategy holds true and the number of scheduled tasks are upper bounded by $\Theta(|\mathcal{K}|)$.
3) When $|\mathcal{K}| > |\mathcal{S}_l|$, then the performance of the DMO strategy is upper bounded by $\Theta(|\mathcal{S}_l|)$, i.e., atmost $|\mathcal{S}_l|$ tasks can be offloaded by the DMO strategy.

Thus, the total performance bound for the proposed DMO strategy is upper bounded by $\min\{\Theta(|\mathcal{K}|), \Theta(|\mathcal{S}_l|)\}$ and the remaining $(|\mathcal{S}_l| - |\mathcal{K}|)$ tasks will wait for next timestamp. A visual representation of our proposed computation offloading strategy is depicted in Fig. 4. ■

Theorem 2: The runtime complexity of the proposed TSCO strategy is $\Theta(\mathcal{S}_l^3)$.

Proof: The complexity of the collaborative computation offloading strategy is divided into two stages. First, the IBTS mechanism classify the task priority for the set of industrial tasks $\mathcal{K}$ with the ranking of IIoT devices in $1 \times \Theta(\mathcal{K}) = \Theta(\mathcal{K})$ time. In the second stage, the DMO mechanism makes offloading decisions. Initially, the time requires to complete the sorting and ranking will take $O(\mathcal{S}_l^2)$ and $\Theta(\mathcal{S}_l)$, respectively. Then, the DMO policy takes constant time to identify the r from the sorted list in step 2. Next, to identify and replace the $\leq r$ values to 0 on matrix E takes $\Theta(\mathcal{S}_l)$ time. Once the matrix is prepared, identifying the offloading task to devices (i.e., step 4) requires $O(\mathcal{S}_l^3)$ time. So, the complexity of the proposed DMO strategy is $O(\mathcal{S}_l^2) + 2 \times \Theta(\mathcal{S}_l) + \Theta(1) + O(\mathcal{S}_l^3) \approx O(\mathcal{S}_l^3)$. Thus, the asymptotic complexity of our proposed TSCO strategy becomes $\Theta(\mathcal{K}) + \Theta(\mathcal{S}_l^3) \approx \Theta(\mathcal{S}_l^3)$. ■

Theorem 3: For a given industrial fog network with a speedup factor $\mathscr{U}$ and time critical parameters $\Delta_{kl}^{\text{delay}}$ and $\Delta_{kl}^{\text{energy}}$, the computation offloading decisions must follow the conditions $\Delta_m > \max(\Delta_{kl}^{\text{delay}}, \Delta_{kl}^{\text{energy}})$ and $(\Delta_{kl}^{\text{delay}}/\Delta_{kl}^{\text{energy}}) < 1$, where $\Delta_{kl}^{\text{delay}}$ and $\Delta_{kl}^{\text{energy}}$ denote the coefficients of delay time and energy time utilities.

Proof: Specifically, offloading time is the sum of communication and computation time on remote processing devices, and it should be less than the execution time on IIoT devices to improve performance as shown in Fig. 5. Thus, in order to save execution time, it is preferable to offload computation data to the fog devices or cloud server, when local execution meets condition $\mathbb{T}_{ka}^{\text{process}} > \mathbb{T}_{al}^{\text{process}} + \mathbb{T}_{al}^{\text{up}}$. Similarly, when a computation data fits the energy criteria $\mathscr{P}_a^{\text{process}}\mathbb{T}_{ka}^{\text{process}} > \mathcal{H}_i\mathbb{T}_{al}^{\text{process}} + \mathscr{P}_l^{\text{up}}\mathbb{T}_{al}^{\text{up}}$, it is worth offloading to consider remote processing than running tasks locally, where $l \in \mathcal{S}_l$. Therefore, offloading can save energy when

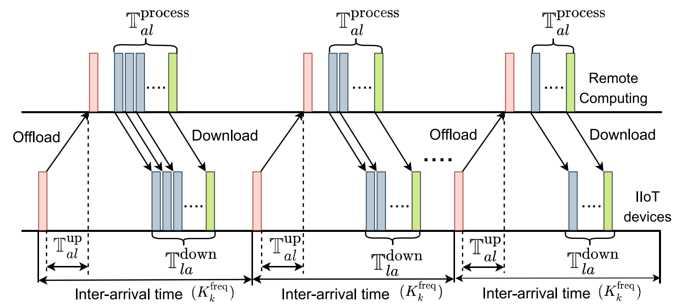


Fig. 5. Traffic attributes of fog networks.

the energy spent on remote communication and processing is less than the energy consumed by the IIoT device. Let $\mathbb{T}_{ka}^{\text{process}} = \mathscr{U}\mathbb{T}_{al}^{\text{process}}$, $1 < \mathscr{U} < \mathscr{U}_{\max}$, where $\mathscr{U}$ denotes the speedup factor for remote processing devices. Now, we can rewrite the above two conditions as follows:

$$\mathbb{T}_{ka}^{\text{process}} > \mathbb{T}_{ka}^{\text{process}}/\mathscr{U} + \mathbb{T}_{al}^{\text{up}} \tag{24}$$

$$\mathscr{P}_a^{\text{process}}\mathbb{T}_{ka}^{\text{process}} > \mathcal{H}_i\mathbb{T}_{ka}^{\text{process}}/\mathscr{U} + \mathscr{P}_l^{\text{up}}\mathbb{T}_{al}^{\text{up}} \tag{25}$$

where $l \in \mathcal{S}_l$. The inequalities in (24) and (25) impose large $\mathscr{U}$ for the server, small data size and large transmission bandwidth. According to [25], we can derive (24) and (25) with two time critical values $\Delta_{kl}^{\text{delay}}$ and $\Delta_{kl}^{\text{energy}}$ as follows:

$$\Delta_{kl}^{\text{delay}} = \frac{\Delta_{kl}^{\text{delay}}}{\mathscr{U}} + \mathbb{T}_{al}^{\text{up}} \Rightarrow \Delta_{kl}^{\text{delay}} = \frac{\mathbb{T}_{al}^{\text{up}}}{1 - 1/\mathscr{U}} \tag{26}$$

$$\mathscr{P}_a^{\text{process}}\Delta_{kl}^{\text{energy}} > \mathcal{H}_i\Delta_{kl}^{\text{energy}}/\mathscr{U} + \mathscr{P}_l^{\text{up}}\mathbb{T}_{al}^{\text{up}}$$
$$\Rightarrow \Delta_{kl}^{\text{energy}} = \mathscr{P}_l^{\text{up}}\mathbb{T}_{al}^{\text{up}}/\mathscr{P}_a^{\text{process}} - \mathcal{H}_i/\mathscr{U}. \tag{27}$$

It comes from the fact that (26) and (27) strongly requites $1 - (1/\mathscr{U}) > 0$ and $\mathscr{U} > (\mathcal{H}_i/\mathscr{P}_a^{\text{process}})$. Especially, when $\mathcal{H}_i = \mathscr{P}_l^{\text{up}}$, inequality in (25) reduced to

$$\mathbb{T}_{ka}^{\text{process}} > \frac{\mathcal{H}_i}{\mathscr{P}_a^{\text{process}}}\left(\frac{\mathbb{T}_{ka}^{\text{process}}}{\mathscr{U}} + \mathbb{T}_{al}^{\text{up}}\right). \tag{28}$$

Therefore, in order to minimize execution delay while extending battery life, $\mathbb{T}_{ka}^{\text{process}}$ must fulfill the following criteria $\Delta_m > \max(\Delta_{kl}^{\text{delay}}, \Delta_{kl}^{\text{energy}})$, which satisfy the original requirement. Besides to associate $\Delta_{kl}^{\text{delay}}$ and $\Delta_{kl}^{\text{energy}}$, let

$$\frac{\Delta_{kl}^{\text{delay}}}{\Delta_{kl}^{\text{energy}}} = \frac{\mathbb{T}_{al}^{\text{up}}}{1 - \frac{1}{\mathscr{U}}} \cdot \frac{\mathscr{P}_a^{\text{process}} - \frac{\mathcal{H}_i}{\mathscr{U}}}{\mathscr{P}_l^{\text{up}}\mathbb{T}_{al}^{\text{up}}} < 1$$

which further qualifies the second requirement and this completes the proof of Theorem 3. ■

Theorem 4: The proposed TSCO computation offloading problem is NP-hard.

Proof: The proposed TSCO algorithm decides the assignment of IIoT tasks to appropriate computing devices (edge/fog/cloud). To proceed this, the TSCO algorithm needs to minimize (17a). So, the proposed computation offloading algorithm is considered a generalized assignment problem of the optimization problem for minimizing the overall performance overhead of the system. It is known that the generalized assignment and optimization problems are NP-hard. Hence, our proposed TSCO strategy is also NP-hard and is a particular case of assignment problem. ■

IV. Numerical Analysis

In this section, we investigate and compare the efficiency of our proposed computation offloading strategy with two standard baseline algorithms, such as random computation offloading (RCO) and priority-based computation offloading (PBCO) in terms of: 1) *processing delay*; 2) *energy consumption*; and 3) *throughput*. Moreover, we consider three recent works, such as DPTO [26], DDPT [3], and EECO [9] for illustrating the performance improvement of our proposed strategy. A concise outline of the existing baseline algorithms is explained below.

1) *RCO Strategy:* The RCO strategy randomly schedules the tasks and offloads them to nearby computing devices without considering the importance of task scheduling.
2) *PBCO Strategy:* In the PBCO strategy, tasks are scheduled according to device priority. However, offload the tasks without considering the computational capability of the remote processing devices.
3) *DDPT Strategy:* The DDPT strategy prioritizes the tasks based on queue allocation strategy and offloads the tasks, utilizing a graph matching theorem with the objective of optimizing computation delay.
4) *EECO Strategy:* The EECO strategy mainly prioritizes the tasks and schedules them using a stochastic optimization technique. Finally, a constraint-restricted offloading method is for optimizing energy–delay over the network.
5) *DPTO Strategy:* In the DPTO strategy, tasks are classified according to a heuristic process. Then, a multilevel feedback queue is used to schedule the tasks. Finally, a heuristic technique for making task-offloading decisions.

Essentially, these algorithms operate as a reference to determine the performance enhancement of our TSCO strategy in the IoT–fog–cloud networks.

A. Simulation Setup

The complete simulation is done on Intel Core i7-2600 CPU @3.40 GHz $\times$ 8 with 8-GB RAM using *Ubuntu* operating system. We consider 100 IIoT sensors that generate real-time tasks with $\mathcal{K}_k^{\text{in}} = [5, 20]$ MB in the fog networks. We consider $\lambda_k = [10, 20]$ task/s, $K_k^{\text{freq}} = [10, 20]$ s and $\mathscr{B} = 40$ MBps. Furthermore, we set $\varphi = 1900$ [cycles/byte], $\mathcal{M} = 10, \mathcal{N} = 2$, $\mathcal{A} = 20$, and $\mathcal{X}(k, l) = \{0, 1\}$ [27]. To capture the dynamicity and make the environment more functional, we assign the CPU threshold $\Gamma_l^{\max}$, energy threshold $\mathcal{E}_l^{\max}$, and delay threshold $\mathcal{T}_l^{\max}$ within the maximum limit. We consider $\Gamma_a^{\text{CPU}} << \Gamma_m^{\text{CPU}}$ and $\Gamma_m^{\text{CPU}} << \Gamma_n^{\text{CPU}}$ throughout the experiment. The initial task distribution following conditions $\Gamma_a^{\text{CPU}} > K_k^{\text{CPU}}$ and $\Gamma_a^{\text{CPU}} \leq K_k^{\text{CPU}}$ are presented in Fig. 6. Other simulation parameters are obtained from [3] and [9], respectively. Table III lists numerous standard parameters used in the simulation.

B. Processing Delay

This metric represents the total amount of time $\mathbb{T}_{kl}^{\text{total}}$ taken for executing a task $k \in \mathcal{K}$ on various computing devices $\mathcal{S}_l$, including IIoT devices $\mathcal{A}$, fog devices $\mathcal{M}$, and cloud servers $\mathcal{N}$. However, performing a task k in the fog device $m \in \mathcal{M}$ or cloud server $n \in \mathcal{N}$ includes additional delay for data transmission $\mathbb{T}^{\text{up}}$ and result fetching $\mathbb{T}^{\text{down}}$ to the system. From $\mathbb{T}_{am}^{\text{up}} = \mathcal{X}(k, m) \times \varphi.\mathcal{K}_k^{\text{in}} / \mathscr{R}_{am}^{\text{up}}$ and $\mathbb{T}_{ma}^{\text{down}} = \mathcal{X}(m, k) \times \mathcal{K}_k^{\text{out}} / \mathscr{R}_{ma}^{\text{down}}$, it can be easily observed that transmission delay for a task $k \in \mathcal{K}$ mostly depends on several network parameters, such as available transmission bandwidth $\mathscr{B}$, channel power gain $\mathscr{H}_a$, transmission power $\mathscr{P}^{\text{up}}$, etc. However, processing delay $\mathbb{T}^{\text{process}}$ mostly depends on input data size $\mathcal{K}_k^{\text{in}}$ and computational frequency Γ^{CPU} of the computing device. From $\mathbb{T}_{am}^{\text{process}} = \mathcal{X}(k, m).\varphi.\mathcal{K}_k^{\text{in}} / \Gamma_m^{\text{CPU}}$ we can observe that as the input size $\mathcal{K}_k^{\text{in}}$ increases, processing delay also increases. However, processing delay can be optimized by increasing the CPU frequency Γ^{CPU} of the computing devices, as the processing delay inversely proportional to the processing CPU frequency of the computing device $m \in \mathcal{M}$, i.e., $\mathbb{T}_{am}^{\text{process}} \propto 1/\Gamma_m^{\text{CPU}}$. Fig. 7(a) illustrates the analysis of normalized processing delay on various computing devices, whereas Fig. 7(b) depicts the comparative study of processing delay

TABLE III
Parameters Used in Experimental Analysis

Parameters	Values	Parameters	Values
$\mathcal{K}$	100	$\mathcal{N}$	2
$\mathcal{M}$	10	$\mathcal{A}$	20
$\mathcal{X}(k,l)$	{0,1}	λ_k	[10,20]
$\mathscr{B}^{\text{up}}$	40 MHz	$\mathscr{B}^{\text{down}}$	40 MHz
$\mathcal{K}_k^{\text{in}}$	2-20 MB	Υ	[0.2,0.5]
$\mathscr{P}_a^{\text{process}}$	$\times$1.2	$\mathcal{E}_l^{max}$	5 unit
$\mathcal{T}_l^{max}$	6 unit	Γ_a^{CPU}	10×10^6
Γ_n^{CPU}	10×10^8	Γ_m^{CPU}	100×10^9

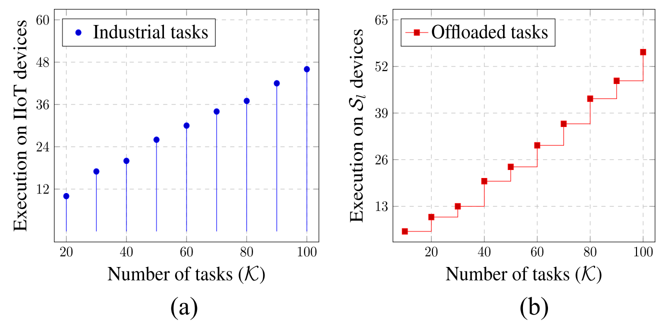


Fig. 6. Initial task execution strategy in various computing devices: (a) IoT devices $\Gamma_a^{\text{CPU}} > K_k^{\text{CPU}}$ and (b) virtual computing devices $\Gamma_a^{\text{CPU}} \leq K_k^{\text{CPU}}$.

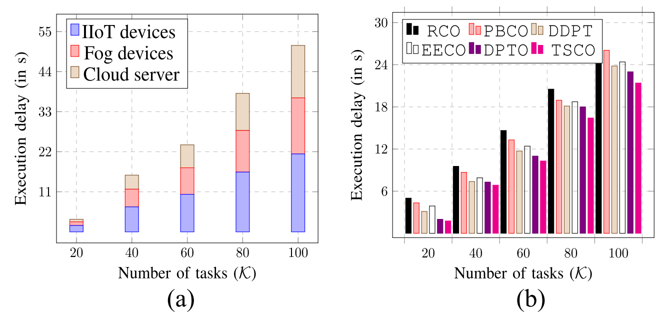
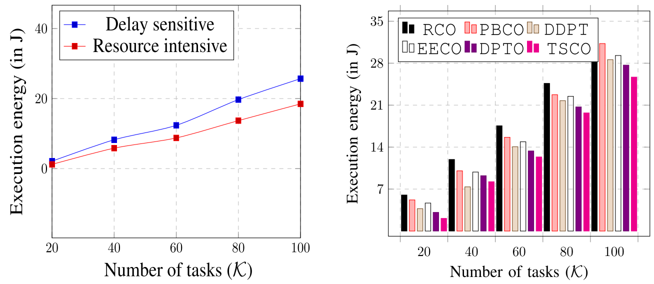
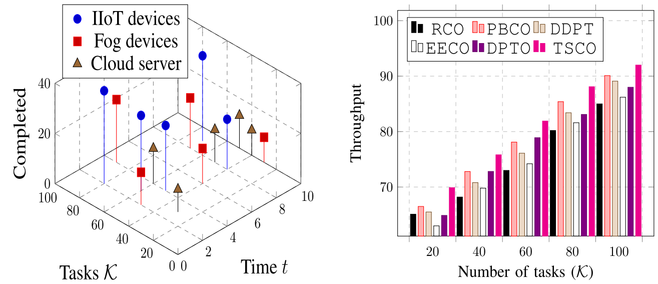


Fig. 7. Analysis of processing delay $(\mathbb{T}_{kl}^{\text{total}})$: (a) on various executing devices and (b) comparison with the existing algorithms.



(a) (b)

Fig. 8. Analysis of energy consumption $(\mathbb{E}_{kl}^{\text{total}})$: (a) on various executing devices and (b) comparison with the existing algorithms.



(a) (b)

Fig. 9. Performance analysis of throughput: (a) on various executing devices and (b) comparison with the existing algorithms.

with existing algorithms. It is obvious to say that the proposed TSCO strategy achieves better performance than RCO, PBCO, DDPT, DDPT, and EECO algorithms.

C. Energy Consumption

Energy utilization for a task $k \in \mathcal{K}$ can be regarded as the amount of energy used $\mathbb{E}_{kl}^{\text{total}}$ to process a task, which includes transmission energy $\mathbb{E}_a^{\text{up}}$, processing energy $\mathbb{E}_a^{\text{process}}$, and downloading energy $\mathbb{E}_a^{\text{down}}$. For easy implementation, we omit the energy consumption rate for waiting tasks in the execution queue on computing devices $\mathcal{S}_l$. Equation (16) indicates that energy consumption rate $\mathbb{E}^{\text{process}}$ on IIoT devices directly proportional to the processing capability $1/\Gamma^{\text{CPU}}$ of computing devices, i.e., $\mathbb{E}^{\text{process}} \propto 1/\Gamma^{\text{CPU}}$. Moreover, total energy consumption rate $\mathbb{E}_{kl}^{\text{total}}$ also increases with the increase in input data size $\mathcal{K}_k^{\text{in}}$ and decreases with the CPU frequency Γ^{CPU}. However, we can regulate the energy consumption by increasing the transmission bandwidth $\mathscr{B}$ and CPU frequency Γ^{CPU} of the computing devices. Fig. 8(a) demonstrates the review of approximate processing energy consumption on various computing devices with $\mathbb{E}_l^{\text{CPU}} = 1.2$ unit, and Fig. 8(b) represents the comparative analysis of energy consumption with existing algorithms, which is better than 22%, 21%, 18%, and 19% compared with RCO, PBCO, DDPT, DDPT, and EECO algorithms. The reason is that the proposed TSCO strategy offloads resource-hungry tasks to the cloud server, thus better utilization of energy consumption for fog devices.

D. Throughput

This parameter represents another level of performance evaluation for satisfying energy $\mathcal{E}_l^{\max}$ and delays $\mathcal{T}_l^{\max}$ constraints, i.e., how many numbers of tasks $\mathcal{K}$ complete their execution within the given threshold bound. Our proposed computation offloading technique offload delay and energy bound tasks to the nearby fog devices $\mathcal{M}$ based on their priority index and offload rest of the resource-hungry and low-priority index tasks to the centralized cloud data center $\mathcal{N}$ for execution. Fig. 9(a) and (b) represents the number of various priorities of tasks that completes execution. It is noteworthy to see from Fig. 9(a) that IIoT executable tasks complete its execution within the given bound, but in some cases, other tasks fail to satisfy execution deadline due to limited capacity in fog devices. The performance analysis of throughput is also exhibited in Fig. 9(b). It is clear to analyze from Fig. 9(b) that the proposed TSCO strategy achieves a reasonable performance for executing the maximum number of tasks than other existing RCO, PBCO, DDPT, DDPT, and EECO algorithms while satisfying several constraints.

V. Conclusion

In this article, we introduced a hierarchical computation offloading technique called TSCO, by collaborating and utilizing both fog and cloud resources simultaneously. First, we defined our objective function as the joint optimization of weighted energy-latency consumption, while satisfying several QoS constraints. To solve this optimization problem, we developed an index-based transmission scheduling strategy to reduce the computation overhead from the IIoT devices. Then, our proposed mixed-linear-programming-based computation offloading method offloads the tasks based on the importance and makes a near-optimal decision to select suitable computing devices. Extensive simulation results exhibited the effectiveness of the proposed TSCO strategy over standard algorithms in terms of average waiting time 20%–26% and average energy consumption rate 12%–17%, respectively. In the future, we will enhance our proposed computation offloading strategy for optimizing various user-oriented Quality of Experience using deep reinforcement learning in the distributed environment.

References

[1] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[2] Q. Wang, S. Guo, J. Liu, and Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing," *Sustain. Comput. Informat. Syst.*, vol. 21, pp. 154–164, Mar. 2019.

[3] I. Sarkar, M. Adhikari, N. Kumar, and S. Kumar, "Dynamic task placement for deadline-aware IoT applications in federated fog networks," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1469–1478, Jan. 2022.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[5] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4702–4711, Oct. 2018.

[6] J. Wang, K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 9, pp. 2062–2075, Sep. 2020.

[7] Z. Zhou, Y. Guo, Y. He, X. Zhao, and W. M. Bazzi, "Access control and resource allocation for M2M communications in industrial automation," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 3093–3103, May 2019.
[8] B. Sudharsan *et al.*, "Toward distributed, global, deep learning using IoT devices," *IEEE Internet Comput.*, vol. 25, no. 3, pp. 6–12, May/Jun. 2021.
[9] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Joint computation offloading and scheduling optimization of IoT applications in fog networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 3266–3278, Oct.–Dec. 2020.
[10] Y. Ding, C. Liu, X. Zhou, Z. Liu, and Z. Tang, "A code-oriented partitioning computation offloading strategy for multiple users and multiple mobile edge computing servers," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4800–4810, Jul. 2020.
[11] M. Mukherjee *et al.*, "Latency-driven parallel task data offloading in fog computing networks for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6050–6058, Sep. 2020.
[12] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1524–1537, Mar. 2020.
[13] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14198–14211, Dec. 2020.
[14] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y.-C. Tian, "Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing," *IEEE Access*, vol. 6, pp. 55923–55936, 2018.
[15] R. Yadav *et al.* "Smart healthcare: RL-based task offloading scheme for edge-enable sensor networks," *IEEE Sensors J.*, vol. 21, no. 22, pp. 24910–24918, Nov. 2021.
[16] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Stackelberg game for service deployment of IoT-enabled applications in 6G-aware fog networks," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5185–5193, Apr. 2021.
[17] Q. Li, S. Wang, A. Zhou, X. Ma, F. Yang, and A. X. Liu, "QoS driven task offloading with statistical guarantee in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 278–290, Jan. 2022.
[18] V. Karagiannis, P. A. Frangoudis, S. Dustdar, and S. Schulte, "Context-aware routing in fog computing systems," *IEEE Trans. Cloud Comput.*, early access, Aug. 6, 2021, doi: 10.1109/TCC.2021.3102996.
[19] C. Avasalcai, B. Zarrin, and S. Dustdar, "EdgeFlow -developing and deploying latency-sensitive IoT edge applications," *IEEE Internet Things J.*, early access, Jul. 30, 2021, doi: 10.1109/JIOT.2021.3101449.
[20] I. Sarkar, M. Adhikari, N. Kumar, and S. Kumar, "A collaborative computational offloading strategy for latency-sensitive applications in fog networks," *IEEE Internet Things J.*, early access, Aug. 12, 2021, doi: 10.1109/JIOT.2021.3104324.
[21] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Collaborative AI-enabled intelligent partial service provisioning in green industrial fog networks," *IEEE Internet Things J.*, early access, Sep. 7, 2021, doi: 10.1109/JIOT.2021.3110910.
[22] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Intelligent service deployment policy for next-generation industrial edge networks," *IEEE Trans. Netw. Sci. Eng.*, early access, Oct. 27, 2021, doi: 10.1109/TNSE.2021.3122178.
[23] C. E. Shannon, "Communication in the presence of noise," *Proc. IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949.
[24] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Trans. Mobile Comput.*, vol. 19, no. 1, pp. 29–43, Jan. 2020.
[25] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, 2013, pp. 728–732.
[26] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5773–5782, Jul. 2020.
[27] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, "Joint task offloading and resource allocation for delay-sensitive fog networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, 2019, pp. 1–7.

Abhishek Hazra (Student Member, IEEE) received the bachelor's degree from the National Institute of Technology Agartala, Agartala, India, in 2014, and the master's degree in computer science and engineering from the National Institute of Technology Manipur, Imphal, India, in 2018. He is currently pursuing the Ph.D. degree with the Indian Institute of Technology (Indian School of Mines) Dhanbad, Dhanbad, India.

He has authored and coauthored various national and international journal and conference articles. His research areas of interests are in the field of fog computing, 6G, machine learning, and Industrial Internet of Things.

Praveen Kumar Donta (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines) Dhanbad, Dhanbad, India, in June 2021.

He is a University Assistant (Postdoc) with the Research Division of Distributed Systems, Vienna University of Technology, Vienna, Austria. He was a visiting Ph.D. student for six months during Ph.D. with the Mobile & Cloud Lab, University of Tartu, Tartu, Estonia. His current research is in machine learning for WSNs, IoT, and fog/edge computing.

Dr. Donta is an Editorial Member for *Physical Communication* and *Computer Communications* Journals.

Tarachand Amgoth (Member, IEEE) received the B.Tech. degree in computer science and engineering from Jawaharlal Nehru Technological University Hyderabad, Hyderabad, India, in 2002, the M.Tech. degree in computer science engineering from the National Institute of Technology Rourkela, Rourkela, India, in 2006, and the Ph.D. degree from the Indian Institute of Technology (Indian School of Mines) [IIT (ISM)] Dhanbad, Dhanbad, India, in 2015.

He is currently working as an Associate Professor with the Department of Computer Science and Engineering, IIT (ISM) Dhanbad. His current research interests include fog/edge computing, and Internet of Things.

Schahram Dustdar (Fellow, IEEE) received the master's and Ph.D. degrees from Johannes Kepler Universität Linz, Linz, Austria, in 1989 and 1992, respectively.

He is a Full Professor of Computer Science heading the Research Division of Distributed Systems, Vienna University of Technology, Vienna, Austria.

Prof. Dustdar is a recipient of the ACM Distinguished Scientist Award in 2009, the ACM Distinguished Speaker Award in 2021, and the IBM Faculty Award in 2012, and an Elected Member of the Academia Europaea: The Academy of Europe, where he is the Chairman of the Informatics Section. In 2021, he was elected to the Academy of the United Nations Sciences and Technology Organization. He is the Founding Coeditor-in-Chief of the new *ACM Transactions on Internet of Things* as well as the Editor-in-Chief of *Computing* (Springer). He is an Associate Editor of the IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*, as well as on the Editorial Board of *IEEE Internet Computing* and IEEE Computer Society.