Distributed Service Provisioning with Collaboration of Edge and Cloud in Industry 5.0

Abhishek Hazra, *Member, IEEE,* Alakesh Kalita *Member, IEEE,* and Mohan Gurusamy, *Senior Member, IEEE,*

***Abstract*—Industry 5.0 aims to elevate industrial operations, businesses, and revolution to new heights by promoting sustainable, resilient, and human-centric practices. The popularity of Industry 5.0 is reflected in the increasing demand for real-time and near-edge processing in most latency-critical Industrial Internet of Things (IIoT) applications. However, designing an efficient task priority assignment strategy and accordingly executing tasks within the stipulated deadline is complex and challenging. Therefore, in this work, we design a novel Multi-device Edge Service Provisioning (`MESP`) framework for optimizing delay in Industry 5.0. At first, the `MESP` strategy classifies edge executable tasks using multi-nomial probability theory. Then, we prove that multi-device service demand at the edge devices is an NP-Hard problem, which requires approximate algorithms for finding near-optimal solutions. To follow this, we propose a game-theoretic approach where multiple IIoT devices request various services simultaneously while maximizing their mutual satisfaction. We also examine the structural property of the proposed game and show how this property helps in achieving the equilibrium point of the proposed game with finite improvement steps. Experimental analysis shows that `MESP` reduces computational overhead and end-to-end execution delay by 20-30% compared to standard algorithms.**

***Index Terms*—Industry 5.0, Edge Computing, Distributed Systems, Service Provisioning, Game Theory,**

I. Introduction

WITH the progressive development of Industry 5.0 and the distributed edge computing paradigm, industries are getting modernized and offering satisfactory services to users [1]. Specifically, Industry 5.0 is defined by five unique factors, Artificial Intelligence (AI) which enables systems to learn and make intelligent decisions on their own, Cyber-Physical Systems (CPS) that automate and optimize processes, human-machine interaction to enhance decision-making capabilities, distributed edge computing that facilitates flexibility and quick responsiveness, Big Data for analytics, and Industrial Internet of Things (IIoT) that enables real-time data generation [2]. It is anticipated that the primary use of Industry 5.0 includes enhanced decision-making, predictive maintenance, flexibility, customisation, and safety [3]. Along with technological advancements, Industry 5.0 applications (e.g., supply chain optimization, real-time inventory management, autonomous UAV and industrial robotics) also generate a large volume and velocity of latency critical data such as fire safety alerts, periodic maintenance, faulty warnings, critical events related to high voltage, explosion risk, non-ionizing radiation and security alerts [4]. Besides, all these applications bring several new challenges, including priority-based reliable offloading, multi-device edge service provisioning, optimized end-to-end delay, and priority-based service allocation [5]. Hence, one of the most significant challenges in industrial applications is the reduction of end-to-end latency, emphasizing the importance of achieving lower latency and timely data processing [6]. Similarly, industrial networks require efficient and distributed service provisioning strategies with the collaboration of competent edge-cloud infrastructure to increase the Industry operations' performance.

A. Motivation

Digitization towards Industry 5.0 indicates the need for new latency-critical services and technological advancements in industrial networks. Industrial revolutions rely mainly on edge/cloud-centric architectures to handle dynamic service requests from IIoT devices [7]. However, a sustainable industrial environment demands delay-optimized, resilient, and customer-centric design. In this regard, industrial machines such as connected vehicles, industrial cobots, production lines and assembly machines explicitly produce rapid-race of latency-critical tasks and require on-time reliable processing on near-edge computing devices [8]. However, due to the limited computation capability of IIoT devices and the trade-off between energy and latency for edge/cloud service provisioning, we require a priority-aware task classification strategy to ensure essential tasks can be accomplished near-edge devices and industries can continue to operate [9]. Therefore, our research considers these challenges as motivation and identifies the research gap in creating a latency-critical service provisioning strategy in Industry 5.0 that efficiently executes priority-based IIoT tasks and takes advantage of emerging edge computing while maintaining a number of QoS parameters.

Manuscript received January XX, 2023; revised May XX, 2023; January XX, 2023; accepted February XX, 2024. Date of publication February XX, 2021; date of current version July XX, 2024.

(Corresponding author: Abhishek Hazra.)

A. Hazra is with the Department of Computer Science and Engineering, Indian Institute of Information Technology, Sri City and the Communications & Networks Lab, Department of Electrical and Computer Engineering, National University of Singapore. (e-mail: abhishek.hazra1@gmail.com)

A. Kalita and M. Gurusamy with the Communications & Networks Lab, Department of Electrical and Computer Engineering, National University of Singapore. (e-mail: alakesh.kalita1025@gmail.com, gmohan@nus.edu.sg)

Digital Object Identifier 10.1109/JIOT.2020.3021XXX

B. Related Work

Over the last few years, several research efforts have been made to develop service deployment strategies and automate Industrial processes [10]. However, designing a reliable service provisioning strategy is challenging as Industry 5.0 has its own restrictions, making it difficult for actual deployment. For example, Aazam *et al.* [11] have designed a CoT-based task offloading strategy for sustainable IoT applications. Ghosh *et al.* [12] have developed a 6G-enabled cognitive communication protocol for Industry 5.0. However, these approaches are not suitable for large-scale IIoT deployment. Another possible solution could be using advanced computation and communication technologies for IIoT automation. For example, Adhikari *et al.* have highlighted the need for 6G wireless communication in edge networks. Similarly, Rathee *et al.* [13] have designed a cognitive automation system for making intelligent decisions in IIoT networks. Another critical issue is handling computation overhead while processing delay-sensitive industrial applications [14]. Specifically, computation overhead in Industry 5.0 brings several new challenges and demands an immediate effective problem-solving approach to boost decision-making capabilities.

To solve this challenge, Du *et al.* [15] have designed an Industrial edge computing framework using the Byzantine Machine Learning algorithm in Industry 5.0. Zong *et al.* [16] have developed an end-to-end data transmission protocol for IIoT applications in Industry 5.0. A comparative analysis of existing works is presented in Table I. Even though research has been done from several dimensions and prospects to automate industrial applications, only a few works considered the priority-aware industrial service provisioning issue in industrial networks [17], [18]. Besides, existing efforts didn't pay attention to strategies that involve known payouts or quantifiable consequences of minimizing end-to-end delay in Industry 5.0 [19]. Another critical issue is the mutual satisfaction among industrial devices while obtaining services from remote computing devices. Therefore a collaborative industry-edge-cloud service provisioning with mutual satisfaction among the industrial devices must be needed in order to maintain automation in Industry 5.0.

Thus in this work, we consider two significant challenges in designing an automated multi-device service provisioning strategy, 1) *how to design an efficient task priority assignment strategy that gives priority confidence with probability*. 2) *how to design a multi-device service provisioning strategy that optimizes computation overhead while offering almost mutual satisfaction among industrial devices*. Unlike existing works, we consider a probability-based device selection strategy and a multi-device game theoretic approach for addressing the industrial service provisioning challenges while maintaining execution delay as low as possible. Probability-based device selection techniques enhance resource efficiency by prioritizing devices with a higher probability of success, optimizing resource allocation. In contrast, utilizing a game theory-based service provisioning model, which optimizes resource distribution among devices, is essential for ensuring the efficient utilization of network resources with minimal delays.

TABLE I
COMPARATIVE REVIEW OF EXISTING LITERATURE

Existing Works	Priority Assignment	Collaborative Decision	Low Complexity	Service Provisioning	Mutual Satisfaction
Aazam *et al.* [11]	×	×	✓	✓	×
Du *et al.* [15]	×	✓	✓	×	×
Zong *et al.* [16]	×	×	✓	✓	×
Jain *et al.* [7]	×	×	✓	✓	×
Sarkar *et al.* [20]	×	×	×	✓	×
Mukherjee *et al.* [21]	×	✓	✓	✓	×
Hazra *et al.* [17]	✓	×	✓	✓	×
Otoum *et al.* [22]	✓	×	×	✓	×
Hazra *et al.* [23]	✓	×	×	✓	×
Proposed MESP	✓	✓	✓	✓	✓

C. Contribution

Considering the above-mentioned challenges, in this work, we design a distributed service provisioning framework called Multi-device Edge Service Provisioning (MESP) with the collaboration of edge and cloud in Industry 5.0. The MESP strategy incorporates a probability-based task classification and a game theoretic-based service provisioning method for end-to-end delay optimization while maintaining higher satisfaction among industrial devices. In brief, the major contributions of this paper are listed as follows.

- Design an edge service provisioning framework for maintaining mutual satisfaction and end-to-end processing delay in Industry 5.0. In particular, we design our objective function as a mixed integer nonlinear programming problem with delay as the optimization objective.
- To solve this, we design a novel probability-based task classification strategy to specify a fixed priority for all the IIoT device-generated requests in the industrial networks.
- Then, to achieve the delay-optimized performance, we transform our objective function into a multi-device cooperative game and define its structural property. We also prove that the structural property helps to obtain a nash equilibrium point with finite improvement steps.
- Experimental analysis on various performance metrics validates that our proposed strategy encompasses near-optimal solutions while achieving more promising results as compared with standard baseline algorithms.

The rest of the paper is arranged as follows. In *Section* II, we design our model and formulate the objective function. The proposed MESP methodology is explained in *Section* III. *Section* IV quantifies the performance improvement of our proposed strategy. Finally, the conclusion and scope for future research direction are presented in *Section* V.

II. NETWORK MODEL AND PROBLEM FORMULATION

Considering an industrial edge network with a set of I IIoT devices, denoted by $\mathcal{I} = \{1, 2, \ldots, I\}$, and a set of A resource-oriented edge servers, represented by $\mathcal{A} = \{1, 2, \ldots, A\}$. Besides, we denote $\mathcal{R} = \{1, 2, \ldots, R\}$, be the set of cloud servers located in geo-distributed areas. Let $\mathscr{S}$ denotes the combination of remote computing devices, where $\mathscr{S} = (\mathcal{R} \cup \mathcal{A})$. The $\mathcal{I}$ IIoT devices support mobility and can move in their deployed industrial area at a constant speed $\mathcal{Y}$ [1]. On the other hand, edge servers $\mathcal{A}$ are uniformly distributed in the industrial area to handle excessive execution overhead of the network. Let there be C wireless channels, denoted by $\mathcal{C} = \{1, 2, \ldots, C\}$,

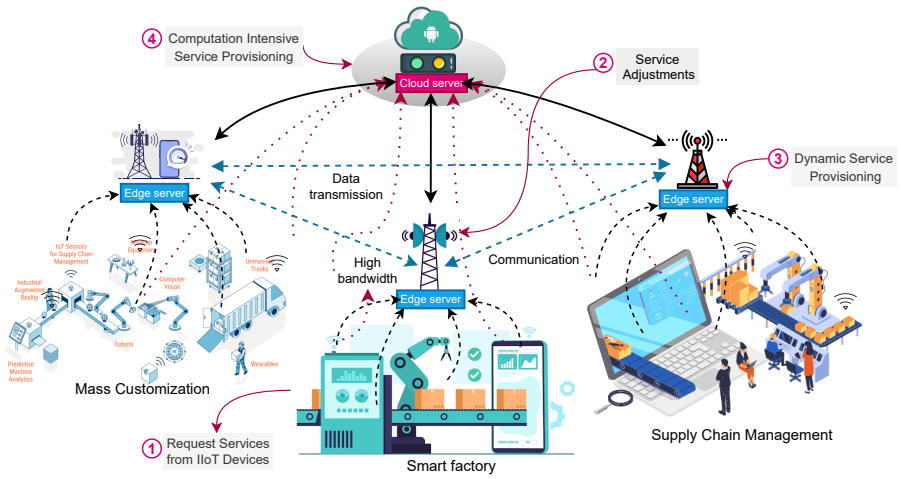


Fig. 1. Edge-enabled Industry 5.0 architecture.

TABLE II
NOTATIONS AND RESPECTIVE MEANINGS

Symbols	Definition
$\mathcal{K}$	Number of industrial tasks in the network
$\mathcal{A}$	Number of edge servers in the industrial network
$\mathcal{R}$	Number of cloud servers in the industrial network
$\mathcal{I}$	Number of IIoT devices in the industrial network
$\mathcal{C}$	Number of wireless channel in the network
$\mathscr{D}_i$	Service provisioning decision profile of a device i
$\Re$	Distance between two computing server
$\mathcal{K}_k^{\text{in}}$	Size of the IIoT generated data k
$\mathcal{K}_k^{\text{CPU}}$	CPU requirement to execute IIoT generated data
$\mathscr{U}_{i,s}^{\text{up}}$	Uplink transmission rate from an IIoT device
$\mathscr{B}_{i,s}^{\text{in}}$	Available data transmission bandwidth
$\mathbf{d}_i$	Service provisioning decision strategy
$\mathbb{T}_{k,i}^{\text{IIoT}}$	Task execution delay on the IIoT device
$\mathbb{T}_{k,s}^{\text{send}}$	Data transmission delay to remote servers
$\mathbb{T}_{k,s}^{\text{server}}$	Service execution delay on the remote server
$\mathscr{C}_s^{\text{server}}$	Computation capacity of the remote computing server
$\mathscr{C}_i^{\text{local}}$	Computation capacity of the local IIoT device

used for communication between the IIoT devices $\mathcal{I}$ and remote servers $\mathscr{S}$. The IIoT devices $\mathcal{I}$ have a set of executable tasks, denoted by $\mathcal{K} = \{1, 2, \ldots, K\}$ and each task can be represented by three key tuples, $\mathcal{K}_k \triangleq \langle \mathcal{K}_k^{\text{in}}, \mathcal{K}_k^{\text{D}}, \mathcal{K}_k^{\text{CPU}} \rangle$, where $\mathcal{K}_k^{\text{in}}$ denotes the input data size, $\mathcal{K}_k^{\text{D}}$ represents the task execution deadline and $\mathcal{K}_k^{\text{CPU}}$ denotes the required computation power (in cycles) to execute the task. Let $\mathscr{D}_i$ be the service provisioning decision profile of a device $i \in \mathcal{I}$, denoted by $\mathscr{D}_i = \{0, 1, \ldots, C\}$ and $\forall \mathbf{d}_i \in \mathscr{D}_i$ refer to its current strategy. Specifically, the decision variable $\mathbf{d}_i$ indicates whether k^{th} task is executed on IIoT device $i \in \mathcal{I}$ or remote server $s \in \mathscr{S}$. Mathematically, we can write $\mathbf{d}_i$ as follows,

$$\mathbf{d}_i = \begin{cases} 1, & \text{if } k\text{th task offloaded to remote server } s \in \mathscr{S} \\ 0, & \text{otherwise.} \end{cases}$$

In this network, each IIoT device $i \in \mathcal{I}$ is connected to several edge servers and can request additional services for optimizing execution overload, as shown in Fig. 1. We have tabulated all the variables in Table II

A. Transmission Model

This section briefly discusses the communication model for the considered industrial edge network. Let $\mathscr{B}_{i,s}^{\text{in}}$ and $\mathscr{O}_i$ represent the minimum bandwidth availability and transmission power of an IIoT device $i \in \mathcal{I}$, respectively. Then, for a given service provisioning decision profile $\mathscr{D}_i$, the uplink transmission rate $\mathscr{U}_{i,s}^{up}$ to a remote server $s \in \mathscr{S}$ can be defined as follows.

$$\mathscr{U}_{i,s}^{\text{up}} = \mathscr{B}_{i,s}^{\text{in}} log_2 \left(1 + \frac{\mathscr{O}_i \mathscr{G}_{i,s}}{w_0 + \sum_{j \in I \setminus \{i\} : \mathbf{d}_j = \mathbf{d}_n} \mathscr{O}_j \mathscr{G}_{j,s}} \right) \quad (1)$$

where, w_0 is the additive white gaussian noise and $\mathscr{G}_{i,s}$ denotes the channel gain between IIoT device $i \in \mathcal{I}$ and remote server $s \in \mathscr{S}$. For uplink transmission rates, channel gain is one of the most important parameters, which describes how strong or powerful the signal is as it travels from the transmitter to the receiver. From the transmission model defined in Eq. (1), we notice that if multiple IIoT devices $\mathcal{I}$ choose the same wireless channel $c \in \mathcal{C}$ to offload their executable tasks, then IIoT devices would experience massive interference on the wireless channel, and it results in lower data rate $\mathscr{U}_{i,s}^{\text{up}}$.

B. Service Execution Model

The service execution model of an industrial edge network can be partitioned into local execution and remote server execution, including edge server and cloud server execution.

1) Local Execution: Initially, IIoT devices $\mathcal{I}$ try to execute their generated tasks $\mathcal{K}$ locally to reduce transmission delay. Let $\mathscr{C}_i^{\text{local}}$ be the computation frequency of an IIoT device $i \in \mathcal{I}$. Then the execution delay $\mathbb{T}_i^{\text{IIoT}}$ to process a task $k \in \mathcal{K}$ on the IIoT device $i \in \mathcal{I}$ can be defined as follows.

$$\mathbb{T}_{k,i}^{\text{IIoT}} = \frac{\mathcal{K}_k^{\text{CPU}}}{\mathscr{C}_i^{\text{local}}} \quad \forall i \in \mathcal{I}, \forall k \in \mathcal{K} \quad (2)$$

2) Remote Execution: However, IIoT devices $\mathcal{I}$ need additional computation assistance from remote servers $\mathscr{S}$ owing to limited computation power $\mathscr{C}_i^{\text{local}}$ and storage capacity. Now

with a strategic data transmission rate $\mathscr{U}_{i,s}^{\text{up}}$, we can define the transmission time $\mathbb{T}_{k,s}^{\text{send}}$ to a remote server $s \in \mathscr{S}$ as follows.

$$\mathbb{T}_{k,s}^{\text{send}} = \frac{\mathcal{K}_k^{\text{in}}}{\mathscr{U}_{i,s}^{\text{up}}} \quad \forall s \in \mathscr{S}, \forall k \in \mathcal{K} \tag{3}$$

Once tasks are obtained, remote servers $\mathscr{S}$ immediately start executing those tasks $\mathcal{K}$. Let $\mathscr{C}_s^{\text{server}}$ be the computation frequency of the remote server. Then the execution delay $\mathbb{T}_{k,s}^{\text{proc}}$ on a remote server $s \in \mathscr{S}$ can be defined as follows.

$$\mathbb{T}_{k,s}^{\text{proc}} = \frac{\mathcal{K}_k^{\text{CPU}}}{\mathscr{C}_s^{\text{server}}} \quad \forall i \in \mathcal{I}, \forall s \in \mathscr{S} \tag{4}$$

Similarly, results of the executed task $\mathcal{K}_k^{\text{in}}$ will be transferred back to the required IIoT device $i \in \mathcal{I}$. Let $\Re$ be the distance between two remote servers and $\mathcal{X}_i^{\text{seg}}$ be the number of hops crossed to reach device i. Then the result downloading time $\mathbb{T}_{k,s}^{\text{result}}$ from remote server $s \in \mathscr{S}$ can be defined as follows.

$$\mathbb{T}_{k,s}^{\text{result}} = \frac{\mathcal{X}_i^{\text{seg}} \Re}{\mathcal{Y}} \quad \forall k \in \mathcal{K}, \forall s \in \mathscr{S} \tag{5}$$

Therefore the remote server execution delay $\mathbb{T}_{k,s}^{\text{server}}$ for a task $k \in \mathcal{K}$ will be the combination of uploading delay $\mathbb{T}_{k,s}^{\text{send}}$, processing delay $\mathbb{T}_{k,s}^{\text{proc}}$ and result fetching delay $\mathbb{T}_{k,s}^{\text{result}}$, defined as follows.

$$\begin{aligned} \mathbb{T}_{k,s}^{\text{server}} &= \mathbb{T}_{k,s}^{\text{send}} + \mathbb{T}_{k,s}^{\text{proc}} + \mathbb{T}_{k,s}^{\text{result}} \\ &= \frac{\mathcal{K}_k^{\text{in}}}{\mathscr{U}_{i,s}^{\text{up}}} + \frac{\mathcal{K}_k^{\text{CPU}}}{\mathscr{C}_s^{\text{server}}} + \frac{\mathcal{X}_i^{\text{seg}} \Re}{\mathcal{Y}} \end{aligned} \tag{6}$$

In this model, we exclude the computation overhead for downloading the results from remote servers, as the result fetching delay is 1/30 times the input data size [17].

C. Problem Formulation

The primary goal of the MEPS approach is to optimize the execution delay for all the IIoT-generated service requests during local execution $\mathbb{T}_{k.i}^{\text{IIoT}}$ and remote server execution $\mathbb{T}_{k,s}^{\text{server}}$, which can be achieved by combining transmission power $\eta = (\mathscr{O}_i)_{\forall i}$, transmission rate $\tau = (\mathscr{U}_{i,s}^{\text{up}})_{\forall i,s}$, service provisioning decision profile $\kappa = (\mathscr{D}_i)_{\forall i}$ and CPU frequency $\varphi = (\mathscr{C}_i^{\text{local}})_{\forall (i)}$. Therefore the total execution delay $\mathbb{T}_k^{\text{total}}$ includes the execution of task $k \in \mathcal{K}$ to either IIoT device $i \in \mathcal{I}$ or remote server $s \in \mathscr{S}$, depending on the priority and requirements of industrial applications.

$$\mathbb{T}_k^{\text{total}} = \underbrace{\mathbb{T}_{k,i}^{\text{IIoT}}\ \mathbb{I}(\mathbf{d}_i, 0)}_{\text{IIoT execution}} + \underbrace{\mathbb{T}_{k,s}^{\text{server}}\ \mathbb{I}(\mathbf{d}_i, c)}_{\text{Remote server execution}} \tag{7}$$

Further, we can derive overall computation overhead $\mathbb{T}^{\text{total}}$ generated from $\mathcal{I}$ industrial devices as $\mathbb{T}^{\text{total}} = \sum_{i \in \mathcal{I}} \mathbb{T}_k^{\text{total}}$. With this formulation, we can mathematically define our objective function and related QoS constraints for designing a delay-efficient distributed service provisioning strategy as.

$$\underset{\kappa, \tau, \varphi, \eta}{\text{minimize}} \quad \sum_{k \in \mathcal{K}} \mathbb{T}_k^{\text{total}} \tag{8a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}} \sum_{s \in \mathscr{S}} \mathscr{C}_i^{\text{local}} << \mathscr{C}_s^{\text{server}},\ \forall s \in \mathscr{S}, \tag{8b}$$

$$\sum_{i \in \mathcal{I}} \mathbf{d}_i = 1,\ \forall i \in \mathcal{I}, \tag{8c}$$

$$\sum_{i \in \mathcal{I}} \mathbf{d}_i \leq |\mathscr{S}|,\ \forall i \in \mathcal{I}, \tag{8d}$$

$$\max\ \left\{ \mathbb{T}_{k,i}^{\text{IIoT}}, \mathbb{T}_{k,s}^{\text{server}} \right\} \leq \mathbb{T}^{\max}, \tag{8e}$$

$$0 \leq \mathcal{K}_k^{\text{CPU}} \leq \mathscr{C}_s^{\text{server}},\ \forall s \in \mathscr{S}, \forall k \in \mathcal{K}, \tag{8f}$$

$$\mathbf{d}_i \in \{0, 1\},\ \forall i \in \mathcal{I} \tag{8g}$$

where $\kappa = \{\mathscr{D}_i |\ \forall i \in \mathcal{I}\}$, $\varphi = \left\{\mathscr{C}_i^{\text{local}} |\ \forall i \in \mathcal{I}\right\}$, $\eta = \{\mathscr{O}_i |\ \forall i \in \mathcal{I}\}$ and $\tau = \left\{\mathscr{U}_{i,s}^{\text{up}} |\ \forall i \in \mathcal{I}, \forall s \in \mathscr{S}\right\}$. Here, constraint (8b) indicates that remote servers $\mathscr{S}$ have higher computation capacity than IIoT devices $\mathcal{I}$. Constraint (8c) and (8d) restricts the maximum service provisioning decision profile upto $|\mathscr{S}|$. Besides, constraint (8e) bounds the task execution delay on various computing devices to $\mathbb{T}^{\max}$. Task $k \in \mathcal{K}$ with a more significant computation need must be processed by a remote server $s \in \mathscr{S}$ and is specified by constraint (8f). Finally, service provisioning decision profile $\mathscr{D}_i$ for an IIoT device $i \in \mathcal{I}$ is defined in constraint (8g).

III. Distributed Service Provisioning

This section derives our multi-device service provisioning strategy named MEPS, with the collaboration of edge and cloud resources in Industry 5.0. The key objective is to reduce the computation burden from industrial networks and optimize the delay overhead for local execution and remote execution. To achieve these objectives, our proposed MEPS strategy follows two steps: *A)Priority Assignment* and *B)Multi-device Service Provisioning*. Before discussing these policies, we prove that the multi-device service provisioning on a single server is NP-Hard and challenging to solve [5], [24].

• **Theorem 1.** *Finding the maximum number of decision profiles $(\mathscr{D}_i)_{\forall i \in \mathcal{I}}$ with multi-device centralized service provisioning is an NP-Hard problem.*

Proof: To prove this, we first introduce the Maximum Cardinality Bin Packing Problem (MCBPP) defined in [24]. Let $\mathscr{A}$ be the set of items with size $\mathcal{U}_i, i \in \mathscr{A}$ and $\mathscr{V}$ be the set of bins with identical capacity $\mathcal{V}$. The MCBPP is to keep the maximum number of items in identical bins $\mathscr{V}$ while satisfying capacity constraint $\sum_{i \in \mathscr{A}} \mathbb{X}_{i,j}\ \mathcal{U}_i \leq \mathcal{V}$. We can also derive MCBPP as $min \sum_{i \in \mathscr{A}} \sum_{j \in \mathscr{V}} \mathbb{X}_{i,j}$, subject to $\sum_{j \in \mathscr{V}} \mathbb{X}_{i,j} \leq 1$, $\sum_{i \in \mathscr{A}} \mathbb{X}_{i,j} \mathcal{U}_i \leq \mathcal{V}$, and $\mathbb{X}_{i,j} \in \{0, 1\}$, *i.e.,* if item $i \in \mathscr{A}$ is assigned to bin $j \in \mathscr{V}$ then $\mathbb{X}_{i,j} = 1$, otherwise $\mathbb{X}_{i,j} = 0$, which is an NP-Hard problem [24]. Now, using [5], we will prove that the multi-device service provisioning strategy is a particular type of MCBPP. Let the number of IIoT devices $\mathcal{I}$ and transmission channels $\mathcal{C}$ be the items $\mathscr{A}$ and bins $\mathscr{V}$ in the MCBPP. Then we can argue that as long as devices utilize the dedicated transmission channel $c \in \mathcal{C}$, the size of the items $\mathcal{U}_i$ will not violate the capacity constraint $\mathcal{V}$.

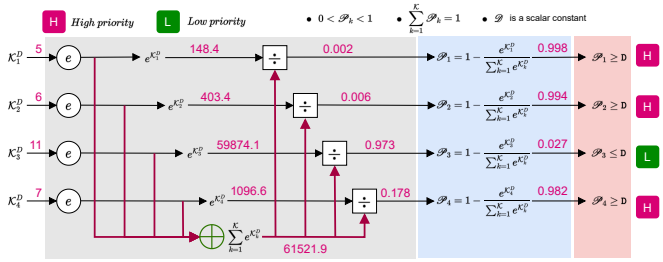


Fig. 2. Illustration of priority assignment strategy.

The modified service provisioning problem can be defined as.

$$\text{maximize} \quad \sum_{i\in\mathscr{A}}\sum_{j\in\mathscr{V}} \mathbb{X}_{i,j} \tag{9a}$$

$$\text{subject to} \quad \sum_{j\in\mathscr{V}} \mathbb{X}_{i,j} \leq 1, \ \forall j \in \mathscr{V}, \tag{9b}$$

$$\mathbb{X}_{i,j} \in \{0,1\}, \ \forall i \in \mathscr{A}, j \in \mathscr{V}, \tag{9c}$$

$$\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s} + \mathscr{O}_i\mathscr{G}_{i,s} \leq \mathcal{V} \tag{9d}$$

Where, $\mathcal{V} = \partial_n + \mathscr{O}_i\mathscr{G}_{i,s}$ and $\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s} \leq \mathcal{V}$. Similarly, as long as IIoT devices $\mathcal{I}$ receive lower interference $\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s}$ from the network, they will try to request additional services from remote servers $s \in \mathscr{S}$. With this, we can prove that if a solution can solve the multi-device service provisioning strategy in constant times, then MCBPP can also be solved in polynomial times, which contradicts the MCBPP statement. Generally, the exhaustion enumeration method can be applied to solve such problems while paying high computation costs. However, game theoretic approaches can leverage this problem with lower computation complexity and offer a high mutual satisfaction solution to participating devices.

A. Priority Assignment

Assigning priority to each industry-generated service request $k \in \mathcal{K}$ is the preprocessing step of our MESP strategy. Initially, industrial devices $\mathcal{I}$ try to gather and process all the tasks $\mathcal{K}$ locally. However, owing to low computation capacity $\mathscr{C}_i^{\text{local}}$ and deadline satisfaction demand $\mathcal{K}_k^{\text{D}}$, IIoT devices request additional computation assistance from nearby computing devices. At first industrial devices $\mathcal{I}$ define the probability distribution of each task $k \in \mathcal{K}$ using the execution deadline $\mathcal{K}_k^{\text{D}}$, determined by $\sigma(\overrightarrow{\mathcal{K}})_k$, where $\overrightarrow{\mathcal{K}}$ is set by the task $k \in \mathcal{K}$. Based on the value of $\sigma(\overrightarrow{\mathcal{K}})_k$, we can determine the priority level of a task $k \in \mathcal{K}$ coming from industrial devices $\mathcal{I}$. Thus, we can define the multinomial probability $\sigma(\overrightarrow{\mathcal{K}})_k$ for a set of tasks $\mathcal{K}$, as follows.

$$\sigma(\overrightarrow{\mathcal{K}})_k = \frac{e^{\mathcal{K}_k^{\text{D}}}}{\sum_{k=1}^{\mathcal{K}} e^{\mathcal{K}_k^{\text{D}}}} \tag{10}$$

Based on the probability distribution $\sigma(\overrightarrow{\mathcal{K}})_k$, we can define the priority index $\mathscr{P}_k$ of each task $k \in \mathcal{K}$ as follows.

$$\mathscr{P}(\overrightarrow{\mathcal{K}})_k = 1 - \sigma(\overrightarrow{\mathcal{K}})_k \tag{11}$$

Now, we can classify a task $k \in \mathcal{K}$ as either a *high priority* or *low priority* with the priority index value $\mathscr{P}(\overrightarrow{\mathcal{K}})_k$ and a predefined threshold D, defined as below.

- **Definition 1:** *A task $k \in \mathcal{K}$ can be classified as highpriority task, if the tasks priority index $\mathscr{P}(\overrightarrow{\mathcal{K}})_k$ is greater than or equal to a predefined threshold D, i.e., $\mathscr{P}(\overrightarrow{\mathcal{K}})_k \geq D$.*
- **Definition 2:** *A task $k \in \mathcal{K}$ can be classified as lowpriority task, if the tasks priority index $\mathscr{P}(\overrightarrow{\mathcal{K}})_k$ is less than a predefined threshold D, i.e., $\mathscr{P}(\overrightarrow{\mathcal{K}})_k < D$.*
- **An illustration example:** Let four IIoT devices $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_4\}$ independently generating tasks with execution deadline $\mathcal{K}_k^D = \{5, 6, 11, 7\}$ sec, as illustrated in Fig. 2. Then using the priority index $\mathscr{P}(\overrightarrow{\mathcal{K}})_k$ defined in Eq. (11), we can calculate $\mathscr{P}(\overrightarrow{\mathcal{K}})_1$ of task $\mathcal{K}_1 \in \mathcal{K}$ as $\mathscr{P}(\overrightarrow{\mathcal{K}})_1 = 1 - (148.4/61521.9) = 0.998$. Similarly, $\mathscr{P}(\overrightarrow{\mathcal{K}})_2 = 1 - (403.4/61521.9) = 0.994$, $\mathscr{P}(\overrightarrow{\mathcal{K}})_3 = 1 - (59874.1/61521.9) = 0.027$ and $\mathscr{P}(\overrightarrow{\mathcal{K}})_4 = 1 - (1096.6/61521.9) = 0.982$. Since the priority index of $\mathscr{P}(\overrightarrow{\mathcal{K}})_1$, $\mathscr{P}(\overrightarrow{\mathcal{K}})_2$ and $\mathscr{P}(\overrightarrow{\mathcal{K}})_4$ are $\geq \text{D} = 0.5$, tasks $\mathcal{K}_1$, $\mathcal{K}_2$ and $\mathcal{K}_4$ will be classified as *high priority* and remaining task $\mathcal{K}_3$ will be identified as *low priority*. *Algorithm* 1 shows all the necessary steps for task classification.

Algorithm 1: *Task Classification*

INPUT: D, $\mathcal{K}_k^{\text{in}}$, $\mathcal{I}$, $\mathcal{K}_k^{\text{D}}$, $\mathcal{K}_k^{\text{CPU}}$,
OUTPUT: *high−priority* tasks, *low−priority* tasks,
for $k = 1$ to $\mathcal{K}$ **do**
  Initialize execution deadline $\mathcal{K}_k^{\text{D}}$;
  Initialize execution threshold D;
  Calculate $\sigma(\overrightarrow{\mathcal{K}})_k = e^{\mathcal{K}_k^D} / \sum_{k=1}^{\mathcal{K}} e^{\mathcal{K}_k^D}$;
  Calculate $\mathscr{P}(\overrightarrow{\mathcal{K}})_k = 1 - \sigma(\overrightarrow{\mathcal{K}})_k$;
  if $\mathscr{P}(\overrightarrow{\mathcal{K}})_k \geq \text{D}$ **then**
    Classify task $k \in \mathcal{K}$ as the *high priority*;
  else
    Classify task $k \in \mathcal{K}$ as the *low priority*;
  end if
end for

B. Multi-device Service Provisioning

After priority assignment, IIoT devices $\mathcal{I}$ request additional services to the remote computing servers $\mathscr{S}$ and transfer data $\mathcal{K}$ through the wireless channel $\mathcal{C}$. However, if IIoT devices $\mathcal{I}$ observe that remote server execution $\mathbb{T}^{\text{server}}$ will be beneficial for optimizing computation overhead $\mathbb{T}^{\text{total}}$, then only devices send service execution request using $\mathbf{d}_i > 0$, otherwise chooses to execute services locally *i.e.,* $\mathbf{d}_i = 0$. Also, when multiple devices ask for more services simultaneously through the same channel $c \in \mathcal{C}$, there may be a chance of elevated interference $\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s}$ on the industrial networks. Therefore, service provisioning decisions $\mathscr{D}_i$ depend not only on the current device's decision interests $\mathbf{d}_i$ but also on the interest of other devices. Compared to existing works, we consider the multi-device wireless channel service provisioning strategy using a game theoretic approach

that ensures consideration of multiple participating devices' interests simultaneously.

1) Game Formulation: The problem of transforming the end-to-end delay optimization objective into a strategic game can be defined by the number of IIoT devices $\mathcal{I}$ and corresponding service provisioning decisions $\mathscr{D}_i$.

• **Definition 3:** *A strategic game is a process of selecting the best decisions $\mathscr{D}_i$, where the outcome depends on not only the behaviour of the current player but also the behaviour of other players.*

Let $\mathbf{d}_{-i} = \{1, 2, \ldots, i-1, i+1, \ldots, I\}$ denote the service provisioning decision profile without device i. If device i knows that other devices made their best decision $\mathbf{d}^*_{-i}$, then device i can take the best decision $\mathbf{d}^*_i$ and choose local execution $\mathbf{d}^*_i = 0$ or remote server execution $\mathbf{d}^*_i > 0$ depending on the outcome of the computation overload. A service provisioning game is also illustrated in Fig. 3. Thus, a game can be formulated to capture the interactions among IIoT devices $\mathcal{I}$. We can rewrite the optimization objective as the minimization of end-to-end execution delay as follows.

$$\underset{\mathbf{d}_i \in \mathscr{D}_i \triangleq \{1,2,\ldots,C\}}{\text{minimize}} \quad \mathbb{T}_i(\mathbf{d}_i, \mathbf{d}_{-i}), \quad \forall i \in \mathcal{I} \tag{12}$$

Now, we define a strategic game with three attributes, $\xi = \langle \mathcal{I}, \{\mathscr{D}_i\}_{\forall i \in \mathcal{I}}, \{\mathbb{T}_k\}_{\forall k \in \mathcal{K}} \rangle$, where $\mathcal{I}$ denotes the set of IIoT players, $\mathscr{D}_i$ denotes the service provisioning decision moves and $\mathbb{T}_k$ denotes the task optimization objective. Specifically, the optimization objective for the strategic game $\mathbb{T}_i(\mathbf{d}_i, \mathbf{d}_{-i})$ can batter be defined as.

$$\mathbb{T}_i(\mathbf{d}_i, \mathbf{d}_{-i}) = \begin{cases} \mathbb{T}^{\text{IIoT}}_{k,i}, & \text{if decision profile } \mathbf{d}_i = 0 \\ \mathbb{T}^{\text{server}}_{k,s}, & \text{if decision profile } \mathbf{d}_i > 0 \end{cases} \tag{13}$$

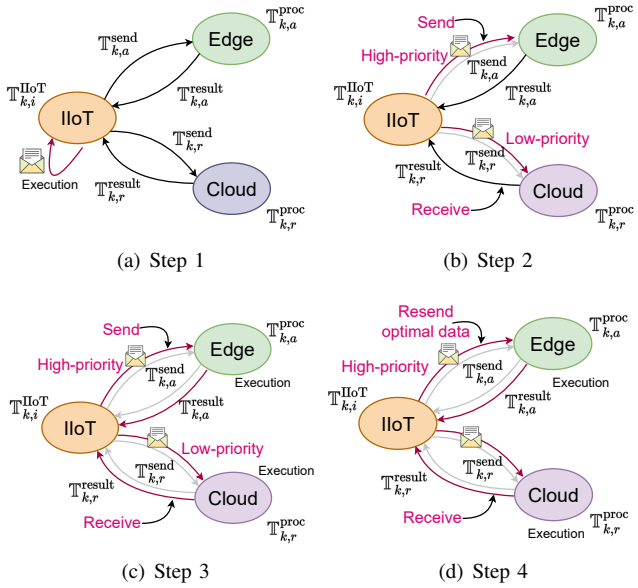
This represents that a device i can always choose to execute its services locally or via a remote computing server. Now we will discuss the concept of the potential game and how a potential game helps to obtain an equilibrium point.

• **Definition 4:** *A game $\xi = \langle \mathcal{I}, \{\mathscr{D}_i\}_{\forall i \in \mathcal{I}}, \{\mathbb{T}_k\}_{\forall k \in \mathcal{K}} \rangle$ is called a potential game if all the participating players $\mathcal{I}$ get a chance to change their decision profile $\boldsymbol{d}_i$ using a global potential function $\Phi(\boldsymbol{d}) : \mathscr{D} \rightarrow \mathbb{R}$, such that $\forall i \in \mathcal{I}$, if $\mathbb{T}_i(\boldsymbol{d}'_i, \boldsymbol{d}_{-i}) < \mathbb{T}_i(\boldsymbol{d}_i, \boldsymbol{d}_{-i})$, then $\Phi_i(\boldsymbol{d}'_i, \boldsymbol{d}_{-i}) < \Phi_i(\boldsymbol{d}_i, \boldsymbol{d}_{-i})$, or $\mathbb{T}_i(\boldsymbol{d}'_i, \boldsymbol{d}_{-i}) > \mathbb{T}_i(\boldsymbol{d}_i, \boldsymbol{d}_{-i})$, then $\Phi_i(\boldsymbol{d}'_i, \boldsymbol{d}_{-i}) > \Phi_i(\boldsymbol{d}_i, \boldsymbol{d}_{-i}), \forall \boldsymbol{d}', \boldsymbol{d} \in \mathscr{D}_i$.*

In other words, if device i receives a worse (improved) computation overhead benefit by unilaterally differing with another strategy, the potential function decreases (increases) with this variation. Two of the most intriguing features of potential games are the existence of Nash equilibrium and the convergence of the learning approach to Nash equilibrium[1].

• **Lemma 1.** *An IIoT device $i \in \mathcal{I}$ can reduce its computation overhead $\mathbb{T}^{total}_k$ with a service provisioning decision profile $\mathscr{D}_i$, if its collected interference $\sum_{j \in I \setminus \{i\}: \boldsymbol{d}_j = \boldsymbol{d}_n} \mathscr{O}_j \mathscr{G}_{j,s}$ satisfies the condition $\sum_{j \in I \setminus \{i\}: \boldsymbol{d}_j = \boldsymbol{d}_n} \mathscr{O}_j \mathscr{G}_{j,s} \leq \frac{\mathscr{O}_i \mathscr{G}_{i,s}}{2^{\frac{\mathcal{K}^{in}_k}{\mathscr{B}^{in}_{i,s}\left(\frac{\mathcal{K}^{CPU}_k}{\mathscr{C}^{local}_i} - \frac{\mathcal{K}^{CPU}_k}{\mathscr{C}^{server}_s}\right)}} - 1} - w_0$.*

[1]https://homepages.cwi.nl/ apt/stra13/potential.pdf



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

Fig. 3. An illustration of service provisioning game, (a) Initial IIoT execution strategy. (b) Request additional services from the remote servers (c) Send *high priority* tasks to the edge server and *low priority* tasks to the cloud server. (d) If there is high interference in the wireless channel, send the optimal amount of data to the edge server.

Proof: From the formulation Eq. (2) and Eq. (6), we observe that remote execution delay is much higher than local execution and can be written as.

$$\mathbb{T}^{\text{IIoT}}_{k,i} \geq \mathbb{T}^{\text{server}}_{k,s} \tag{14}$$

We can simplify Eq. (14) as follows.

$$\mathcal{K}^{\text{CPU}}_k / \mathscr{C}^{\text{local}}_i - \mathcal{K}^{\text{CPU}}_k / \mathscr{C}^{\text{server}}_s \geq \mathcal{K}^{\text{in}}_k / \mathscr{U}^{\text{up}}_{i,s} \tag{15}$$

Putting the value of $\mathscr{U}^{\text{up}}_{i,s}$ into the Eq. (15) yields.

$$\mathcal{K}^{\text{CPU}}_k / \mathscr{C}^{\text{local}}_i - \mathcal{K}^{\text{CPU}}_k / \mathscr{C}^{\text{server}}_s \geq \mathcal{K}^{\text{in}}_k / \mathscr{B}^{\text{in}}_{i,s} log_2 \left(1 + \frac{\mathscr{O}_i \mathscr{G}_{i,s}}{w_0 + \sum_{j \in I \setminus \{i\}: \mathbf{d}_j = \mathbf{d}_n} \mathscr{O}_j \mathscr{G}_{j,s}}\right) \tag{16}$$

$$log_2 \left(1 + \frac{\mathscr{O}_i \mathscr{G}_{i,s}}{w_0 + \sum_{j \in I \setminus \{i\}: \mathbf{d}_j = \mathbf{d}_n} \mathscr{O}_j \mathscr{G}_{j,s}}\right) \geq \mathcal{K}^{\text{in}}_k / \mathscr{B}^{\text{in}}_{i,s} \left(\mathcal{K}^{\text{CPU}}_k / \mathscr{C}^{\text{local}}_i - \mathcal{K}^{\text{CPU}}_k / \mathscr{C}^{\text{server}}_s\right) \tag{17}$$

Now rearranging both sides, we have.

$$1 + \frac{\mathscr{O}_i \mathscr{G}_{i,s}}{w_0 + \sum_{j \in I \setminus \{i\}: \mathbf{d}_j = \mathbf{d}_n} \mathscr{O}_j \mathscr{G}_{j,s}} \geq 2^{\frac{\mathcal{K}^{in}_k}{\mathscr{B}^{in}_{i,s}\left(\frac{\mathcal{K}^{CPU}_k}{\mathscr{C}^{local}_i} - \frac{\mathcal{K}^{CPU}_k}{\mathscr{C}^{server}_s}\right)}} \tag{18}$$

$$\frac{\mathscr{O}_i \mathscr{G}_{i,s}}{w_0 + \sum_{j \in I \setminus \{i\}: \mathbf{d}_j = \mathbf{d}_n} \mathscr{O}_j \mathscr{G}_{j,s}} \geq 2^{\frac{\mathcal{K}^{in}_k}{\mathscr{B}^{in}_{i,s}\left(\frac{\mathcal{K}^{CPU}_k}{\mathscr{C}^{local}_i} - \frac{\mathcal{K}^{CPU}_k}{\mathscr{C}^{server}_s}\right)}} - 1 \tag{19}$$

$$\frac{\mathscr{O}_i \mathscr{G}_{i,s}}{2^{\frac{\mathcal{K}^{in}_k}{\mathscr{B}^{in}_{i,s}\left(\frac{\mathcal{K}^{CPU}_k}{\mathscr{C}^{local}_i} - \frac{\mathcal{K}^{CPU}_k}{\mathscr{C}^{server}_s}\right)}} - 1} \geq w_0 + \sum_{j \in I \setminus \{i\}: \mathbf{d}_j = \mathbf{d}_n} \mathscr{O}_j \mathscr{G}_{j,s} \tag{20}$$

$$\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s} \leq \frac{\mathscr{O}_i\mathscr{G}_{i,s}}{\frac{\mathcal{K}_k^{\text{in}}}{2^{\frac{\mathscr{B}_{i,s}^{\text{in}}\left(\frac{\mathcal{K}_k^{\text{CPU}}}{\mathscr{C}_i^{\text{local}}}-\frac{\mathcal{K}_k^{\text{CPU}}}{\mathscr{C}_s^{\text{server}}}\right)}{\mathcal{K}_k^{\text{in}}}}-1}} - w_0 \quad (21)$$

From Eq. (21), we can easily observe that when an IIoT device receives lower interference $\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s}$, then it is always beneficial to transfer computation data to the remote edge server for execution. Otherwise, the system will incur a high delay penalty for remote execution. Using Lemma 1, we indeed prove that multi-device service provisioning is an exact potential game and all the participating devices follow a potential function defined in [5], $\Phi(\mathbf{d}) = \frac{1}{2}\sum_{i=1}^{\mathcal{I}}\sum_{j=1,j\neq i}^{\mathcal{I}} \mathscr{O}_i\mathscr{G}_{i,s}\mathscr{O}_j\mathscr{G}_{j,s}\mathscr{D}_{\{\mathbf{d}_i=\mathbf{d}_j\}}\mathscr{D}_{\{\mathbf{d}_i>0\}} + \sum_{i=1}^{N}\mathscr{O}_i\mathscr{G}_{i,s}T_i\mathscr{D}_{\{\mathbf{d}_i=0\}}$.

• **Corollary 1:** *A decision profile $\boldsymbol{d} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_{\mathcal{I}}\}$ in a multi-device service provisioning game ξ eventually converges to an equilibrium state if no device $i \in \mathcal{I}$ can get a further chance to change its decisions $\boldsymbol{d}_i$.*

• **Corollary 2:** *The equilibrium point in a strategic game ξ always gives an optimal decision profile $\boldsymbol{d}^* = \{\boldsymbol{d}_1^*, \boldsymbol{d}_2^*, \ldots, \boldsymbol{d}_{\mathcal{I}}^*\}$ to participating devices $\mathcal{I}$. If devices $\mathcal{I}$ further get a chance to improve their decision $\boldsymbol{d}^*$, then that point will also be the equilibrium point, considering devices will change their decisions $\boldsymbol{d}^*$ only if there is scope for performance benefit.*

• **Corollary 3:** *When the participating devices $\mathcal{I}$ are in a potential game and make optimal decisions $\boldsymbol{d}^* = \{\boldsymbol{d}_1^*, \boldsymbol{d}_2^*, \ldots, \boldsymbol{d}_{\mathcal{I}}^*\}$, then they will always gain increased mutual satisfaction with finite improvement steps, even if any change requires in their decisions $\boldsymbol{d}^*$.*

2) Distributed Service Provisioning: The key objective of designing the distributed service provisioning strategy is to obtain maximum mutual satisfaction among the industrial devices $\mathcal{I}$ before execution. This satisfaction rate can be obtained by utilizing the structural property of the devices with finite improvement steps. Let the industrial edge networks follow a $\mathcal{T}$ time-slated system, $\mathcal{T} = \{1, 2, \ldots, T\}, \forall t \in \mathcal{T}$, where the difference between two timeslots t and $t+1$ can be defined by Δt. At first, IIoT devices $\mathcal{I}$ measure the channel interference $\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s}$ for data transmission. Then based on the received interference $\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s}$, devices calculate their best response Θ_i using finite improvement property, defined as $\Theta_i \triangleq \{\widetilde{\mathbf{d}}_i : \widetilde{\mathbf{d}}_i = arg\min_{\mathbf{d}_i\in\mathscr{D}_i} \mathbb{T}_i(\mathbf{d}, \mathbf{d}_i(t)) \text{ and } \mathbb{T}_i(\widetilde{\mathbf{d}}_i, \mathbf{d}_i(t)) < \mathbb{T}(\mathbf{d}_i(t), \mathbf{d}_i(t))\}$. The devices that receive nonzero value, *i.e.,* $\Theta_i \neq \{\}$ will get a chance to improve their decision Θ_i in the next time instance Δ_{t+1}. Otherwise, devices send an optimal amount of data $\mathcal{K}_k^{\text{optimal}}$ in time Δ_{t+1} and continue the whole process.

• **Theorem 2.** *The optimal amount of service requests $\mathcal{K}_k^{\text{optimal}}$, $\forall k \in \mathcal{K}$ to be transferred to an edge server $s \in \mathscr{S}$ is $\mathcal{K}_k^{\text{optimal}} = \mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}}/\mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}} + \mathcal{K}_k^{\text{in}}$.*

Proof : When a device $i \in \mathcal{I}$ decides to unload its computation data $\mathcal{K}_i^{\text{in}}$ to an edge server $s \in \mathscr{S}$, it tries to offload optimal data size for reducing remote execution delay and improve performance speedup. As the execution delay is an important concern for industrial networks, we can determine the IIoT execution delay is same as the edge server execution delay as $\mathbb{T}_{k,i}^{\text{IIoT}} = \mathbb{T}_{k,s}^{\text{send}} + \mathbb{T}_{k,s}^{\text{proc}}$. By putting the values of $\mathbb{T}_{k,s}^{\text{send}}$ and $\mathbb{T}_{k,s}^{\text{proc}}$ we can define remote execution as $\mathbb{T}_{k,s}^{\text{send}} + \mathbb{T}_{k,s}^{\text{proc}} = \{\mathcal{K}_k^{\text{in}}/\mathscr{U}_{i,s}^{\text{up}} + \mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}}/\mathscr{C}_s^{\text{server}}\}\mathbf{d}_{i,s} = \beta_{i,s}\mathcal{K}_k^{\text{in}}$, where, $\beta_{i,s} = \mathbf{d}_{i,k}/\mathscr{U}_{i,s}^{\text{up}} + \mathcal{K}_k^{\text{CPU}}\mathbf{d}_{i,s}/\mathscr{C}_s^{\text{server}}$.

Let $\mathcal{K}_k^{\text{optimal}}$ denotes the optimal amount of task offloaded to an edge server $s \in \mathscr{S}$, where $0 \leq \mathcal{K}_k^{\text{optimal}} \leq \mathcal{K}_k^{\text{in}}$. Then replacing $\mathcal{K}_k^{\text{in}}$ by $\mathcal{K}_k^{\text{optimal}}$ in $\{\mathcal{K}_k^{\text{in}}/\mathscr{U}_{i,s}^{\text{up}} + \mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}}/\mathscr{C}_s^{\text{server}}\}\mathbf{d}_{i,s}$ we obtain

$$\mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}}/\mathscr{C}_i^{\text{local}} = \beta_{i,s}\mathcal{K}_k^{\text{optimal}} \quad (22)$$

Now adding the IIoT executable tasks yields.

$$\mathcal{K}_k^{\text{CPU}}\left(\mathcal{K}_k^{\text{in}} - \mathcal{K}_k^{\text{optimal}}\right)/\mathscr{C}_i^{\text{local}} = \beta_{i,s}\mathcal{K}_k^{\text{optimal}} \quad (23)$$

From Eq. (23), we can derive the optimal amount of tasks offloaded to the edge server.

$$\mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}}/\mathscr{C}_i^{\text{local}} - \mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{optimal}}/\mathscr{C}_i^{\text{local}} = \beta_{i,s}\mathcal{K}_k^{\text{optimal}} \quad (24)$$

Rearranging both sides yields.

$$\mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}}/\mathscr{C}_i^{\text{local}} = \mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{optimal}}/\mathscr{C}_i^{\text{local}} + \beta_{i,s}\mathcal{K}_k^{\text{optimal}} \quad (25)$$

$$\mathcal{K}_k^{\text{optimal}} = \left(\mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}}/\mathcal{K}_k^{\text{CPU}}\mathcal{K}_k^{\text{in}} + \mathcal{K}_k^{\text{in}}\right) \quad (26)$$

Thus, by sending $\mathcal{K}_k^{\text{optimal}}$ amount of service request to the edge server $s \in \mathscr{S}$, industrial devices $\mathcal{I}$ can easily reduce the execution overhead $\mathbb{T}_{k,s}^{\text{server}}$ from the network. With this strategy, multiple devices $\mathcal{I}$ can update their decision profile Θ_i simultaneously and eliminate their computation overhead $\mathbb{T}^{\text{total}}$. The complete steps of the distributed service provisioning strategy are presented in *Algorithm* 2.

• **Theorem 3.** *The computational complexity of the proposed service provisioning strategy is $\mathcal{O}(o) + \mathcal{O}(\widetilde{o}(q + \widetilde{o} \times p))$, where $o = |\mathcal{I}|$, $p = |\mathcal{A}|$ and $q = |\mathcal{R}|$.*

Algorithm 2: Distributed Service Provisioning

INPUT: *high priority* tasks, *low priority* tasks, $\mathcal{I}$, $\mathcal{R}$, $\mathcal{A}$, $\mathcal{C}$, $\mathscr{D}_i$, $\mathscr{O}_i$, $\mathscr{G}_{i,s}$, $\mathscr{B}_{i,s}^{\text{in}}$, $\mathscr{C}_i^{\text{local}}$, $\mathscr{C}_s^{\text{server}}$,
OUTPUT: Computation overhead $\mathbb{T}_k^{\text{total}}$,
Initialize $\mathscr{G}_{i,s}$, $\mathscr{B}_{i,s}^{\text{in}}$ and $\mathbf{d}_i = 0$;
for $k = 1$ to $\mathcal{K}$ **do**
  Send *low priority* tasks to $r \in \mathcal{R}$;
  for all the *high priority* tasks **do**
    Select a wireless channel $c \in \mathcal{C}$;
    Calculate interference $\sum_{j\in I\setminus\{i\}:\mathbf{d}_j=\mathbf{d}_n} \mathscr{O}_j\mathscr{G}_{j,s}$;
    Calculate $\Theta_i \triangleq \{\widetilde{\mathbf{d}}_i : \widetilde{\mathbf{d}}_i = arg\min_{\mathbf{d}_i\in\mathscr{D}} \mathbb{T}_i(\mathbf{d}, \mathbf{d}_i(t))$ and $\mathbb{T}_i(\widetilde{\mathbf{d}}_i, \mathbf{d}_i(t)) < \mathbb{T}(\mathbf{d}_i(t), \mathbf{d}_i(t))\}$;
    if response $\Theta_i \neq \{\}$ **then**
      Update decision $\mathbf{d}_i$ at next instance Δ_t;
      Send *high priority* tasks to server $a \in \mathcal{A}$;
      Calculate $\mathbb{T}_{k,a}^{\text{server}} = \mathbb{T}_{k,a}^{\text{send}} + \mathbb{T}_{k,a}^{\text{proc}} + \mathbb{T}_{k,a}^{\text{result}}$;
    else
      Send optimal tasks $\mathcal{K}_k^{\text{optimal}}$ to server $r \in \mathcal{R}$;
      Calculate $\mathbb{T}_{k,r}^{\text{server}} = \mathbb{T}_{k,r}^{\text{send}} + \mathbb{T}_{k,r}^{\text{proc}} + \mathbb{T}_{k,r}^{\text{result}}$;
    end if
    Return computation overhead $\mathbb{T}^{\text{total}}$;
  end for
end for

TABLE III
PARAMETERS WITH RESPECTIVE VALUES

Parameters	Values	Parameters	Values
$\mathcal{A}$	[10,100]	$\mathcal{F}$	[5,10]
$\mathcal{C}$	3	$\mathscr{B}_{i,s}^{\text{in}}$	20MHz
$\mathcal{K}_k^{\text{in}}$	5MB	$\mathscr{O}_i$	100mW
$\mathscr{C}_i^{\text{local}}$	4×10^7	$\mathscr{C}_a^{\text{server}}$	40×10^9
$\mathscr{C}_r^{\text{server}}$	100×10^{10}		

Proof : The computation complexity of the proposed MESP strategy is twofold. Let, $m = |\mathcal{I}|$, $p = |\mathcal{A}|$ and $q = |\mathcal{R}|$. In the first phase, IIoT devices check their CPU and memory capacity for local execution. This step takes $1 \times \mathcal{O}(o)$ times. For the remaining tasks $\widetilde{o}$, $1 \leq \widetilde{o} \leq o$, IIoT devices define priority level in $1 \times \mathcal{O}(\widetilde{o})$. In the second phase, *low priority* service requests are sent to cloud servers for execution, which may take $\mathcal{O}(\widetilde{o}) \times \mathcal{O}(q)$ times. On the other hand, *high priority* services are transferred to edge servers. This process takes almost $\mathcal{O}(\widetilde{o}) \times \mathcal{O}(p)$ times. However, owing to high interference and low beneficial edge execution, IIoT devices resend optimal-size data to the edge servers. In the worst case, this retransmission takes atmost $\mathcal{O}(\widetilde{o})^2 \times \mathcal{O}(p)$ time. Thus, the overall computation complexity becomes $\mathcal{O}(o) + \mathcal{O}(\widetilde{o} \times q) + \mathcal{O}(\widetilde{o}^2 \times p)$, which can be further simplifies to $\mathcal{O}(o) + \mathcal{O}(\widetilde{o}(q + \widetilde{o} \times p))$.

IV. EXPERIMENTAL ANALYSIS

In this section, we numerically analyze the performance of the proposed MESP strategy in terms of computation overhead, end-to-end delay, and computation complexity. To present the superiority, we compared our proposed strategy with two baseline algorithms named Edge-Centric Execution (ECE) and Cloud-Centric Execution (CCE), discussed as follows.

- **Edge-Centric Execution (ECE):** With this strategy, IIoT devices offload tasks to the default edge server without considering the CPU capacity of that server.
- **Cloud-Centric Execution (CCE):** This strategy sends all the IIoT-generated tasks to resource-rich cloud servers for execution and analysis. Though the CCE technique supports scalability but incurs high computation costs.

Further, we have considered three state-of-the-art techniques, Deadline-based Dynamic Task Placement (DDPT) [25], Delay-Aware Service Provisioning (DASP) [26], and Parallel Task Offloading and Content Caching (PTOCC) [27] for comparative analysis. The virtue of our proposed scheme is also validated using a statistical test called ANOVA, which illustrates a statistical significance among the participating algorithms.

A. Simulation Setup

For experimental analysis, we use Python programming language on Intel® Core™ i7-2600 CPU @ 3.40GHz × 8 system. In the simulation, we consider $\mathcal{I} = 100$ randomly generating service requests over industrial networks. We set $\mathcal{A} = 10$ and $\mathcal{R} = 3$ to handle the computation overhead of the network. Further, we fix channel gain as 4, $\mathscr{B}_{i,s}^{\text{in}} = 20$MHz, $\mathcal{K}_k^{\text{in}} = 5$MB (at most), transmission power as 100mW, and noise as -100dBm [1]. Besides, we consider processing density as 1900 [cycle/byte], minimum CPU capacity of IIoT devices as 4×10^7, edge servers as 40×10^9, and cloud servers as 100×10^{10}. To capture the network dynamicity, we set threshold values to their maximum limit. Other simulation parameters are considered from [23] and [17]. Besides, we specify $\mathscr{C}_i^{\text{local}} << \mathscr{C}_a^{\text{server}}$ and $\mathscr{C}_i^{\text{local}} << \mathscr{C}_r^{\text{server}}$ throughout the experiment. It can be observed that several crucial factors can affect the performance of the algorithm, including task characteristics, network conditions, and device capabilities. The right balance between these factors affects the efficiency of the MESP system. A Summary of the parameters is listed in Table III.

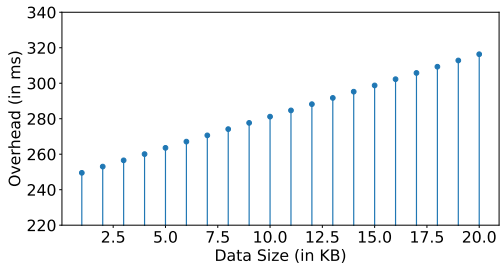


Fig. 4. MEPS performance evaluation in terms of computation overhead variation with respect to input data size $\mathcal{K}_k^{\text{in}}$.

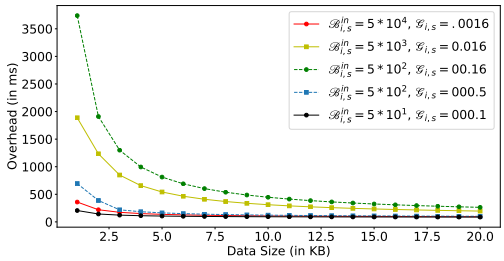


Fig. 5. MEPS performance evaluation in terms of computation overhead variation with respect to transmission bandwidth $\mathscr{B}_{i,s}^{\text{in}}$.

B. Impact of Computation Overhead

The computation overhead $\mathbb{T}^{\text{total}}$ in a network represents any computation expenses related to execution, resource scheduling, sending, receiving, and other overheads needed to execute a service request. It is an essential factor for the industrial decision-making process, *i.e.,* whether to consider a particular feature or not. To evaluate the computation overhead of our model, we consider a range of input data sizes $\mathcal{K}_k^{\text{in}}$ with high transmission bandwidth $\mathscr{B}_{i,s}^{\text{in}}$. Fig. 4 shows the variation of computation overhead with the increase in IIoT devices. However, computation overhead $\mathbb{T}^{\text{total}}$ can be restrained by increasing the number of computing servers $\mathscr{S}$ and sending optimal amounts of data $\mathcal{K}_k^{\text{optimal}}$ to those servers for execution, as illustrated in Fig. 5. On the other hand, increasing transmission bandwidth and computation capacity of the edge and cloud servers also helps optimize computation overhead.

C. End-to-End Service Delay

This metric defines the overall task execution delay for all the IIoT-generated tasks $\mathcal{K}$ on remote computing devices

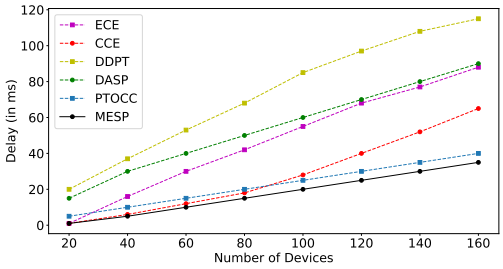


Fig. 6. MEPS performance evaluation in terms of end-to-end delay.

$\mathscr{S}$, *i.e.,* total end-to-end delay $\mathbb{T}_k^{\text{total}}$ includes task uploading delay $\mathbb{T}_{k,s}^{\text{send}}$, remote server execution delay $\mathbb{T}_{k,s}^{\text{proc}}$, and result fetching delay $\mathbb{T}_{k,s}^{\text{result}}$. From $\mathbb{T}_{k,i}^{\text{IIoT}}\,\mathbb{I}(\mathbf{d}_i,0)+\mathbb{T}_{k,s}^{\text{server}}\,\mathbb{I}(\mathbf{d}_i,c)$, we can observe that the total execution delay $\mathbb{T}_k^{\text{total}}$ depends on service provisioning decision profile $\mathscr{D}_i$ and can further be controlled by increasing the transmission power $\mathscr{O}$ of the IIoT devices. Fig. 6 illustrates the comparative end-to-end delay analysis between our proposed strategy and standard baseline algorithms. It can easily be observed from the figure that our proposed algorithm has the capability to optimize end-to-end delay to 23% compared with ECE, CCE, DDPT, DASP, and PTOCC strategies. This is due to the decision-making simplicity and selection of optimal hyper-parameter settings for remote execution. Thus overall execution delay also decreases. It can also be noted that the PTOCC strategy has almost similar performance to MESP strategy when the number of devices and corresponding IIoT data is less. However, the overall end-to-end delay increases with the number of devices.

D. Complexity Analysis

The standard ECE strategy incurs a time complexity of $\mathcal{O}(o+\widetilde{o}p\mathcal{K})$, as it entails offloading all tasks to the edge devices. Conversely, the CCE strategy demands more time, $\mathcal{O}(o+\widetilde{o}q\mathcal{K})$, in comparison to ECE. This is primarily due to CCE offloading all tasks to the cloud server, encompassing both uploading and downloading delays. On the other hand, the DDPT strategy employs a deadline-based device selection approach for offloading tasks to edge devices and cloud servers. It exhibits a time complexity of $\mathcal{O}(o+op+oq)$. Furthermore, the DASP strategy, which adopts an edge-centric computation offloading strategy, renders task offloading decisions within $\mathcal{O}(o+\widetilde{o}p+\widetilde{o}q)$ time. Similarly, the PTOCC strategy is characterized by its latency-driven and parallel task execution mechanism with a time complexity of $\mathcal{O}(o)+\mathcal{O}(\widetilde{o}(q+\widetilde{o}p))$ for task execution. Conversely, our proposed strategy MESP intelligently assesses the network dynamics and strategically selects a suitable computing device for task offloading based on mutual satisfaction. This results in notably lower computational complexity, specifically as $\mathcal{O}(o)+\mathcal{O}(\widetilde{o}(q+\widetilde{o}p))$. Table IV shows the complexity analysis among various techniques.

E. Statistical Analysis

To analyze the performance of our MEPS framework, we consider a hypothesis test on the results obtained through numerical analysis called ANOVA. Analysis of variance (ANOVA) is used to estimate discrepancies among performing groups $\mathscr{G}_n$. Specifically, we evaluate a one-way ANOVA test among the groups to indicate whether a null hypothesis can be rejected or not. Making alternative hypothesis $(\mathscr{H}_a)$ and null hypothesis $(\mathscr{H}_0)$ are essential for the ANOVA test. The null hypothesis establishes equivalency among the respective groups, *i.e.,* $\mathscr{H}_0 : \{\mathscr{G}_1^\pi = \mathscr{G}_2^\pi = \cdots = \mathscr{G}_n^\pi\}$, where $i \in \mathscr{G}_n$. At the same time, the alternative hypothesis considered at least one discrepancy among the individual group pairs, *i.e.,* $\mathscr{H}_a$: $\left\{\exists\left(\mathscr{G}_i^\pi, \mathscr{G}_j^\pi\right) : \mathscr{G}_i^\pi \neq \mathscr{G}_j^\pi\right\}$, where $\mathscr{G}_i^\pi \in \{\mathscr{G}_1^\pi, \mathscr{G}_2^\pi, \ldots \mathscr{G}_3^\pi\}$.

TABLE IV
ANALYSIS OF COMPLEXITY WITH EXISTING ALGORITHMS

Scheme	Complexity
Standard ECE	$\mathcal{O}(o+\widetilde{o}p\mathcal{K})$
Standard CCE	$\mathcal{O}(o+\widetilde{o}q\mathcal{K})$
DDPT [25]	$\mathcal{O}(o+op+oq)$
DASP [26]	$\mathcal{O}(o+\widetilde{o}p+\widetilde{o}q)$
PTOCC [27]	$\mathcal{O}(o+\widetilde{o}pq)$
Proposed MESP	$\mathcal{O}(o)+\mathcal{O}(\widetilde{o}(q+\widetilde{o}p))$

TABLE V
ANOVA TEST FOR PERFORMANCE METRICS

Method	Source of Variation	SS	*df*	MS	*F*	*P value*	*F critical*
Computation Overhead	Between Groups	7770786	3	2590262	7.5673	0.0042	3.4902
	Within Groups	4107556	12	342296.3	-	-	-
	Total	11878343	15	-		-	-
End-to-End Delay	Between Groups	3553.68	3	1184.56	6.9620	0.0057	3.4902
	Within Groups	2041.75	12	170.1453	-	-	-
	Total	5595.43	15	-	-	-	-

To analyze the statistical performance, we consider identical group means and $\alpha\ level = 0.05$ among the performing groups. We have also analyzed existing algorithms with various performance matrices, such as computation overhead and end-to-end delay. Here, α defines the conflicting H_0 hypothesis, and $(1-\alpha)$ indicates the probability of approving H_0 hypothesis, also known as Confidence Interval (CI). CI defines a range of values where the group uses 95% CI. The resultant analysis of the ANOVA test is $F\ statistic$. It is worth noting that if $P\ value < \alpha\ level$ and $F\ statistic < F\ critic$, then a null hypothesis can be rejected. Table V also illustrates that $P\ value$ of computation overhead (0.0042) and end-to-end delay (0.0057) are less compared to $\alpha\ level$. Delivers more confidence for not accepting H_0 hypothesis *i.e.,* the mean of DDPT, DASP, PTOCC, and MESP for various performance metrics, are unequal. We have also depicted the confidence intervals for various computational overheads, as shown in Fig. 7.

V. CONCLUSION

This paper proposed an efficient service provisioning strategy called MESP for minimizing end-to-end delay and computation overhead in Industry 5.0. The key intuition is to promote industrial automation while enhancing mutual satisfaction among IIoT devices. We defined our objective function as the mixed-integer delay minimization problem subject to

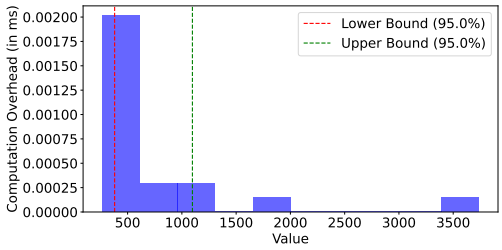


Fig. 7. Analysis of CI for various computation overhead

satisfying constraints. To solve this, MESP classifies edge executable tasks using multi-nomial probability theory on IIoT devices. Then a multi-device service provisioning game is formulated. MESP helps to send multiple service requests to remote servers while maintaining the structural property of the game. Experimental analysis of various performance metrics validates the effectiveness of our proposed strategy compared to existing algorithms.

REFERENCES

[1] Y. Liu, S. Wang, J. Huang, and F. Yang, "A Computation Offloading Algorithm Based on Game Theory for Vehicular Edge Networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[2] P. K. R. Maddikunta, Q.-V. Pham, B. Prabadevi, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.

[3] H. Liu, Y. Sun, J. Cao, S. Chen, N. Pan, Y. Dai, and D. Pan, "Study on UAV Parallel Planning System for Transmission Line Project Acceptance Under the Background of Industry 5.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5537–5546, 2022.

[4] 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.

[5] 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.

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

[7] D. K. Jain, Y. Li, M. J. Er, Q. Xin, D. Gupta, and K. Shankar, "Enabling Unmanned Aerial Vehicle Borne Secure Communication With Classification Framework for Industry 5.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5477–5484, 2022.

[8] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Fog Computing for Energy-Efficient Data Offloading of IoT Applications in Industrial Sensor Networks," *IEEE Sensors Journal*, vol. 22, no. 9, pp. 8663–8671, 2022.

[9] L. Xu, X. Zhou, Y. Tao, X. Yu, M. Yu, and F. Khan, "AF Relaying Secrecy Performance Prediction for 6G Mobile Communication Networks in Industry 5.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5485–5493, 2022.

[10] S. K. Jagatheesaperumal, M. Rahouti, K. Ahmad, A. Al-Fuqaha, and M. Guizani, "The Duo of Artificial Intelligence and Big Data for Industry 4.0: Applications, Techniques, Challenges, and Future Research Directions," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 12 861–12 885, 2022.

[11] 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.

[12] S. Ghosh, T. Dagiuklas, M. Iqbal, and X. Wang, "A Cognitive Routing Framework for Reliable Communication in IoT for Industry 5.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5446–5457, 2022.

[13] G. Rathee, F. Ahmad, R. Iqbal, and M. Mukherjee, "Cognitive Automation for Smart Decision-Making in Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2152–2159, 2021.

[14] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "A Comprehensive Survey on Interoperability for IIoT: Taxonomy, Standards, and Future Directions," *ACM Comput. Surv.*, vol. 55, no. 1, nov 2021. [Online]. Available: https://doi.org/10.1145/3485130

[15] A. Du, Y. Shen, Q. Zhang, L. Tseng, and M. Aloqaily, "CRACAU: Byzantine Machine Learning Meets Industrial Edge Computing in Industry 5.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5435–5445, 2022.

[16] L. Zong, F. H. Memon, X. Li, H. Wang, and K. Dev, "End-to-End Transmission Control for Cross-Regional Industrial Internet of Things in Industry 5.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4215–4223, 2022.

[17] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Intelligent Service Deployment Policy for Next-Generation Industrial Edge Networks," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 5, pp. 3057–3066, 2022.

[18] X. Chen, "Decentralized Computation Offloading Game for Mobile Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[19] A. Chakraborty and S. Misra, "QoS-Aware Resource Bargaining for Federated Learning Over Edge Networks in Industrial IoT," *IEEE Transactions on Network Science and Engineering*, pp. 1–10, 2022.

[20] I. Sarkar, M. Adhikari, S. Kumar, and V. G. Menon, "Deep Reinforcement Learning for Intelligent Service Provisioning in Software-Defined Industrial Fog Networks," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16 953–16 961, 2022.

[21] M. Mukherjee, S. Kumar, C. X. Mavromoustakis, G. Mastorakis, R. Matam, V. Kumar, and Q. Zhang, "Latency-Driven Parallel Task Data Offloading in Fog Computing Networks for Industrial Applications," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6050–6058, 2020.

[22] S. Otoum, I. A. Ridhawi, and H. Mouftah, "A Federated Learning and Blockchain-enabled Sustainable Energy-Trade at the Edge: A Framework for Industry 4.0," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[23] 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.

[24] J. W. Chinneck, B. Kristjansson, and M. J. Saltzman, *Operations research and cyber-infrastructure*. Springer Science & Business Media, 2009, vol. 47.

[25] I. Sarkar, M. Adhikari, N. Kumar, and S. Kumar, "Dynamic Task Placement for Deadline-Aware IoT Applications in Federated Fog Networks," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1469–1478, 2022.

[26] A. Munusamy, M. Adhikari, M. A. Khan, V. G. Menon, S. N. Srirama, L. T. Alex, and M. R. Khosravi, "Edge-Centric Secure Service Provisioning in IoT-Enabled Maritime Transportation Systems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2021.

[27] Z. Xiao, J. Shu, H. Jiang, J. C. S. Lui, G. Min, J. Liu, and S. Dustdar, "Multi-Objective Parallel Task Offloading and Content Caching in D2D-aided MEC Networks," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2022.