Intelligent Service Deployment Policy for Next-Generation Industrial Edge Networks

Abhishek Hazra, Mainak Adhikari, *Member, IEEE*, Tarachand Amgoth, *Member, IEEE*, and Satish Narayana Srirama, *Senior Member, IEEE*

***Abstract*—Edge computing has appeared as a promising technology for realizing industrial computation data at the edge of the network. The fundamental challenge in edge-enabled industrial networks is how to deploy the service requests while utilizing the available edge resources efficiently. In this paper, we aim to design an intelligent service deployment strategy for simultaneously handling both Industrial Internet of Things (IIoT) generated dynamic service requests and edge resources in the next-generation industrial networks. Initially, we present the objective function as the mixed-integer nonlinear programming problem for optimizing the weighted energy-delay in the edge environment. To accomplish this objective, we model a heuristic-based task execution strategy and exploit the advantage of Deep Reinforcement Learning (DRL) to make accurate decisions in industrial networks. The proposed DRL-based strategy can learn well to control the industrial networks from its own experience and guarantees to handle as many service requests as possible using the set of available resource constraint edge servers. Experimental analysis reveals that the proposed strategy is robust to network changes and achieves better performance than existing algorithms in terms of energy consumption up to 13%, delay minimization by 23%, and other Quality of Service (QoS) parameters.**

***Index Terms*—Service deployment, edge computing, energy efficiency, deep reinforcement learning, industrial networks.**

I. Introduction

DUE to the proliferation of delay-restricted industrial services, the resource (e.g., storage, CPU frequency, and energy) faulty Industrial Internet of Things (IIoT) devices endeavor supplementary support from the edge computing that extends the capability of the centralized cloud servers to the edge of the network [1]. It is obliged that the requested services can be performed within their specific deadline-bound to achieve the best possible outcomes. Industrial edge provides an interconnecting bridge between the IIoT device and edge server by deploying computation servers near the proximity of IIoT devices [2]. However, considering dynamic service demands from several IIoT devices and satisfy delay-deadline constraints at the same time becomes a challenging process for the edge-enabled industrial networks. In several cases, requesting a service without estimating resources directly from the edge server can degrade the Quality of Service (QoS) performance to the IIoT devices. Besides that, the downside of energy and delay restrictions for IIoT devices cannot be neglected [3]. For example, healthcare and mining applications require a prompt response from the processing devices to make immediate decisions [4]. Therefore, the edge server needs a suitable task execution mechanism, and a service deployment strategy to increase the devices satisfaction ratio in the industrial environment.

A. Role of DRL Integration in the Industrial Networks

Next-generation industrial networks are becoming very complex and highly dynamic with millions of smart devices, IoT data, multimedia data, and sensor data, making it difficult to analyze, control, and model [5]. On the other hand, edge-enabled applications undergo several resource utilization, dynamic network access, and efficient service deployment issues while meeting various QoS objectives (e.g., energy, latency, and cost) [6]. Recent developments of the Deep Reinforcement Learning (DRL) technique present an assuring approach that allows industrial networks to control the dynamic system by training the edge environment and learning experience from the environment. By adopting the DRL strategy into the industrial environment, edge servers can easily be transformed into intelligent edge servers and bring several key benefits, including a) making an unbiased decision to effectively process real-time industrial applications on smart IIoT devices or edge servers, b) DRL approach can be used to maintain and control distributed IIoT devices in a complex and scalable manner [7], and c) it can handle extremely complex time-variant industrial environments including changing device states and devices service demands.

B. Purpose of Study

IIoT devices generate a massive volume of real-time data in each time instance for processing and analysis [8]. Besides that, the service requests from different IIoT devices have

Manuscript received October 00, 2021; revised January 00, 2021 and January 69, 2021; accepted February 00, 2021. Date of publication February 00, 2020; date of current version July 00, 2021. This work was supported by DST (SERB), Government of India, under Grant EEQ/2018/000888 and UoH-IoE by MHRD, India (F11/9/2019-U3(A)). The associate editor coordinating the review of this letter and approving it for publication was xxxx xxxx.

(Corresponding author: Tarachand Amgoth.)

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

M. Adhikari is with the department of Computer Science, Indian Institute of Information Technology, Lucknow, India-226002. (e-mail: mainak.ism@gmail.com).

S. N. Srirama is with the School of Computer and Information Sciences, University of Hyderabad, India (e-mail: satish.srirama@uohyd.ac.in).

Digital Object Identifier 10.1109/TNSE.2020.3021XXX

variable sizes and priorities. Due to insufficient computing power, most IIoT devices need to transfer the sensory data to the nearby edge server or centralized cloud servers for further analysis [9]. Such a mechanism introduces a fundamental challenge for on-device task execution and service deployment of IIoT applications for optimizing the energy consumption rate and delay in the industrial networks [10]. Precisely, service deployment denotes the number of tasks to deploy on remote computing devices. On the other hand, energy consumption has become skyrocketing in the industrial networks while computing and communicating IIoT service requests on remote computing devices. Hence, minimizing overall energy consumption while controlling the delay is one of the critical and open challenging issues in the edge environment [11]. Recently, DRL is one of the frequently used algorithms to solve such challenges efficiently and intelligently while observing several network or system parameters. Therefore, the essential research scope of this paper is to model an intelligent service deployment framework for delay critical industrial applications in the edge networks.

C. Contribution

To tackle the above-mentioned challenges, in this paper, we develop a novel service deployment strategy by intercoupling IIoT devices and edge server resources in the edge networks. The primary objective of the proposed strategy is to find an error-free service assignment policy for minimizing the execution delay and overall energy consumption of the applications during data transmission and processing. The main contributions of this paper are listed as follows.

- Aiming to envision an efficient industrial environment, we express the objective function as the sum of weighted energy-latency and introduce several network-wide QoS constraints. In particular, the devices service deployment policy is formulated as a non-linear integer programming problem under the optimized energy and service delay.
- Design a heuristic-based task execution strategy for allocating executable tasks on the IIoT devices. Further, to acquire desired output, we model the problem as the MDP-based state-action-reward problem and adopt the DRL technique for intelligent service deployment of IIoT requests in the edge networks.
- Extensive simulation results confirm that the proposed DRL-based service deployment strategy encompassed near-optimal solutions and presented better results than baseline algorithms over the industrial networks.

The rest of the sections are structured as follows. A summary of the existing related articles is presented in Section II. Section III presents the system model and formulates the objective function with associated constraints. In Section IV, we presented the proposed service deployment policy. Section V depicts the numerical analysis of the proposed technique with various performance metrics. Finally, the conclusion and future research direction are conferred in section VI.

II. Related Work

Over the last few years, various research attempts have been made on edge networks to balance IIoT service requests

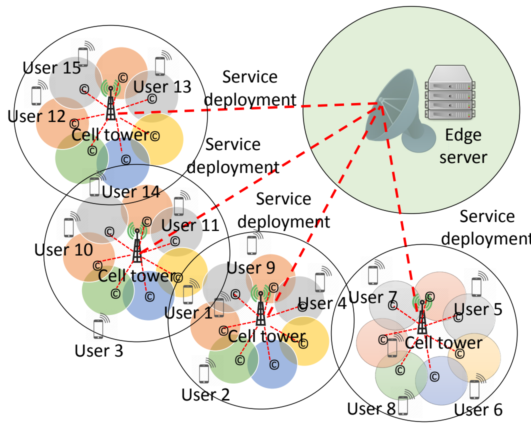


Fig. 1. Illustration of a standard edge computing model.

and edge resources in the distributed industrial environment. In a pioneering work, Xu *et al.* [8] have designed a service offloading framework for utilizing maximum vehicular resources in the industrial networks. Mukherjee *et al.* [12] have investigated a parallel resource provisioning strategy for delay critical industrial applications in a fog environment. In [13], Hazra *et al.* have designed a joint computation and scheduling strategy for handling delay-sensitive IoT applications in fog networks. Iqbal *et al.* [14] have presented a blockchain-based resource sharing and collaborative computing model for the IIoT environment. Similarly, Misra *et al.* [15] have introduced a decentralized task offloading strategy for IoT applications in fog networks. Chen *et al.* [16] have also designed an energy-aware computation offloading algorithm for fog-enabled industrial applications using the dynamic voltage scaling (DVS) technique.

Recently Reinforcement Learning (RL) algorithms have drawn widespread awareness to resolve service offloading and resource provisioning related research issues in academia and industry. For example, in [17], Ale *et al.* have developed an end-to-end DRL framework for optimizing energy and delay in the edge networks. Chen *et al.* [18] have designed an energy-efficient computation offloading scheme for augmented reality applications in edge networks. A DRL-based adaptive task offloading strategy have been developed by Ke *et al.* [10], where the heterogeneous vehicles use the DRL strategy for optimizing energy consumption cost, delay cost, and queue buffer cost in the edge networks. In [19], Sun *et al.* have designed a machine learning approach for optimizing data communication, computation, and migration delay in industrial networks. Recently, in [9] and [4], Goswami *et al.* have introduced AI-enabled resource allocation techniques for secure IIoT networks. Cao *et al.* [20] have also presented a multichannel task offloading mechanism using the multi-agent DRL technique for Industry 4.0 applications.

From the review of the related studies, it can be observed that existing works mainly consider service deployment strategies with various greedy and heuristic techniques [12], [21]. Moreover, very few works consider the identification of IIoT executable tasks and intelligent service deployment strategy combined for complex industrial applications. In specific, DRL techniques are capable of making error-free decisions and simultaneously handle devices dynamic service requests in the unseen industrial environment by optimizing various network and system parameters. In summary, the challenging issues are *"how to design an efficient task execution strategy for identifying executable data on the IIoT devices"*, and *"how to develop an intelligent service deployment framework for delay restricted industrial edge environment"*. The current study complements the use of state-of-the-art DRL techniques with a significant focus on the intelligent decision-making system and optimizing energy consumption rate and processing delay in the industrial networks. A summary of the current related strategies is presented in Table. I.

TABLE I
COMPARATIVE STUDY AMONG THE SERVICE DEPLOYMENT STRATEGIES

Existing works	Task classification	Trade-off analysis	Low complexity framework	Service deployment	Intelligent decisions
[2]	✓	×	×	✓	×
[4]	×	×	✓	✓	✓
[6]	×	×	×	✓	×
[8]	×	×	×	✓	✓
[11]	×	×	✓	✓	✓
[12]	✓	×	✓	✓	×
[13]	✓	×	×	✓	×
[15]	×	×	✓	✓	✓
[20]	×	✓	✓	×	×
[21]	×	×	×	✓	✓
Our work	✓	✓	✓	✓	✓

III. NETWORK MODEL

Considering an edge-enabled industrial networks with a set of IIoT devices $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_I\}$ and a set of edge server, denoted by $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_J\}$, are randomly and uniformly distributed over the edge networks [13], depicted in Fig. 1. Let an IIoT device $\mathcal{M}_i$ executes and simultaneously offloads a set of tasks to the ST, which is equipped with edge server $\mathcal{S}_j$. We assume a service request generated from a IIoT device $\mathcal{M}_i$ with random arrival probability λ_i, can be described as $\mathcal{M}_i = \langle \mathcal{D}_i, \mathcal{O}_i, \mathcal{T}_i^{\max} \rangle$, where $\mathcal{O}_i$ is the requested data size, $\mathcal{D}_i$ is the computation requirement, and $\mathcal{T}_i^{\max}$ is the service processing strict deadline. Let, for a user $\mathcal{M}_i$, $\mathcal{D}_i$ amount of computation cycle is demanded to perform $\mathcal{O}_i$-bit data. Considering a binary service deployment scenario with the assignment vector $\Gamma_i \in \{0\} \cup \{1\}$, where $\Gamma_i = 0$ denotes the local execution, otherwise request services to edge servers $\mathcal{S}$. Ideally, IIoT devices $\mathcal{M}$ strive to process the incoming tasks locally. However, due to the inadequate CPU frequency $(\mathcal{Q}_i^{\text{IIoT}})$ of the IIoT device $\mathcal{M}_i$, the services are forwarded to the remote processing device $\mathcal{S}_j$ (e.g., edge server) that should satisfy minimum latency and resource requirements [17]. For the model uniformity, we consider IIoT devices $\mathcal{M}$ can transmit multiple service requests; however, each service request can be accepted by at most one edge server $\mathcal{S}_j$, i.e., $\sum_{i \in |\mathcal{M}|} \sum_{j \in |\mathcal{S}|} \Gamma_i \leq |\mathcal{S}_j|$. Table II summarizes the key notations used in this article.

A. Local Execution Model

Initially, the set of IIoT devices $\mathcal{M}$ decide to process the incoming tasks on-device. Let $\mathcal{Q}_i^{\text{IIoT}}$ denotes the computation frequency [cycle/second] of a IIoT device $\mathcal{M}_i$. The task

TABLE II
IMPORTANT NOTATIONS

Symbols	Definition
$\mathcal{M}$	Total number of IIoT devices in the edge networks
$\mathcal{S}$	Total number of edge servers in the industrial networks
λ_i	Arrival rate of a service request with random probability
$\mathscr{P}_i$	Transmission power of the IIoT device $\mathcal{M}_i$
$\mathcal{O}_i$	Size of the IIoT generated service request
$\mathcal{D}_i$	Computation requirement of each service request
$\mathcal{T}_i^{\max}$	Execution deadline of each service request
Γ_i	Binary service deployment decision vector for $\mathcal{M}_i$
$\mathcal{Q}_i^{\text{IIoT}}$	Computation frequency of each IIoT device $\mathcal{M}_i$
$\mathcal{Q}_j^{\text{MEC}}$	CPU frequency of the jth processing device
$\mathbb{R}_{ij}$	Data transmission rate for each service request
$\mathcal{D}^{\max}$	CPU-bound on the IIoT devices $\mathcal{M}$
$\mathcal{O}^{\max}$	Memory-bound on the IIoT devices $\mathcal{M}$
$\mathcal{W}_i$	Transmission bandwidth for IIoT devices $\mathcal{M}_i$
$\mathbb{T}_i^{\text{process}}$	Processing delay on the remote computing device $\mathcal{S}_j$
$\mathbb{E}_i^{\text{process}}$	Energy consumption on a remote computing device $\mathcal{S}_j$
$\mathbb{T}_i^{\text{upload}}$	Data transmission time from IIoT device $\mathcal{M}_i$
$\mathbb{E}_i^{\text{upload}}$	Energy consumption for data transmission
$\mathbb{T}_i^{\text{MEC}}$	Total delay in executing a service request on a server $\mathcal{S}_j$
$\mathbb{E}_i^{\text{MEC}}$	Total energy consumption on the edge server $\mathcal{S}_j$

execution delay $\mathbb{T}_i^{\text{IIoT}}$ on the IIoT device $\mathcal{M}_i$ can be expressed as follows.

$$\mathbb{T}_i^{\text{IIoT}} = \frac{\left(1 - \sum_{i \in \mathcal{M}} \Gamma_i\right) \mathcal{O}_i \mathcal{D}_i}{\mathcal{Q}_i^{\text{IIoT}}} \tag{1}$$

Now, we present the energy consumption of a device $\mathcal{M}_i$ as $k\left(\mathcal{Q}_i^{\text{IIoT}}\right)^3$ [22], where k is the chip coefficient of the IIoT device $\mathcal{M}_i$. Thus, the corresponding energy consumption $\mathbb{E}_i^{\text{IIoT}}$ on the IIoT device $\mathcal{M}_i$ is represented as follows.

$$\mathbb{E}_i^{\text{IIoT}} = \left(1 - \sum_{i \in \mathcal{M}} \Gamma_i\right) \mathcal{O}_i \mathcal{D}_i k \left(\mathcal{Q}_i^{\text{IIoT}}\right)^2 \tag{2}$$

B. Service Deployment

IIoT service deployment comprises two phases: uplink data transmission (i.e., through wireless access) and edge server processing. Let $\mathscr{Y}_{i,j}$ be the channel gain between IIoT device $\mathcal{M}_i$ and ST $\mathcal{S}_j$, and $\mathscr{P}_i$ be the consumable transmission power of the IIoT device $\mathcal{M}_i$. According to the Shannon formula, the data transmission rate $\mathbb{R}_{i,j}$ to request a service on the computing device $\mathcal{S}_j$ with noise $\mathscr{D}_i$ can be given by $\mathbb{R}_{i,j} = \mathcal{W}_i log_2 \left(1 + \frac{\mathscr{P}_i \mathscr{Y}_{i,j}}{\mathscr{D}_i}\right)$. Where $\mathcal{W}_i$ be the achievable transmission bandwidth between $\mathcal{M}_i$ and $\mathcal{S}_j$ [6]. Based on $\mathbb{R}_{i,j}$, the uplink transmission time $\mathbb{T}_i^{\text{upload}}$ and transmission energy $\mathbb{E}_i^{\text{upload}}$ to deploy a request on the computing device $\mathcal{S}_j$ can be expressed as follows.

$$\mathbb{T}_i^{\text{upload}} = \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{S}} \frac{\Gamma_i \mathcal{O}_i}{\mathbb{R}_{i,j}} \tag{3}$$

$$\mathbb{E}_i^{\text{upload}} = \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{S}} \frac{\Gamma_i \mathcal{O}_i}{\mathbb{R}_{i,j}} \mathscr{P}_i \tag{4}$$

After receiving, services are being started processing in the edge server. Denote $\mathcal{Q}_j^{\text{MEC}}$ be the computational capacity of the edge server $\mathcal{S}_j$. Then, the task processing delay $\mathbb{T}_i^{\text{process}}$ and execution energy consumption $\mathbb{E}_i^{\text{process}}$ to process the IIoT service requests on edge server $\mathcal{S}_j$ can be represented by.

$$\mathbb{T}_i^{\text{process}} = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{S}} \frac{\Gamma_i \mathcal{O}_i \mathcal{D}_i}{\mathcal{Q}_j^{\text{MEC}}} \tag{5}$$

$$\mathbb{E}_i^{\text{process}} = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{S}} \Gamma_i \mathcal{O}_i \mathcal{D}_i k\big(\mathcal{Q}_j^{\text{MEC}}\big)^2 \tag{6}$$

Throughout this experiment, we consider the industrial edge server $\mathcal{S}_j$ has more innumerable computation frequency than the IIoT device $\mathcal{M}_i$ and service request $\mathcal{D}_i$, i.e., $\mathcal{Q}_i^{\text{IIoT}} << \mathcal{Q}_j^{\text{MEC}}$ and $\mathcal{D}_i << \mathcal{Q}_j^{\text{MEC}}$. Considering a static CPU cycle of device $\mathcal{M}_i$ while deploying a service request on server $\mathcal{S}_j$, then the overall latency $\mathbb{T}_i^{\text{MEC}}$ to process an IIoT service request on the edge server can be expressed as $\mathbb{T}_i^{\text{MEC}} = \mathbb{T}_i^{\text{upload}} + \mathbb{T}_i^{\text{process}} = \sum_{i\in\mathcal{M},j\in\mathcal{S}} \left(\frac{\Gamma_i\mathcal{O}_i}{\mathbb{R}_{i,j}} + \frac{\Gamma_i\mathcal{O}_i\mathcal{D}_i}{\mathcal{Q}_j^{\text{MEC}}}\right)$.

Consequently, the total energy consumption $\mathbb{E}_i^{\text{MEC}}$ to process the same service request will be $\mathbb{E}_i^{\text{MEC}} = \mathbb{E}_i^{\text{upload}} + \mathbb{E}_i^{\text{process}} = \sum_{i\in\mathcal{M},j\in\mathcal{S}} \left(\frac{\Gamma_i\mathcal{O}_i}{\mathbb{R}_{i,j}}\mathscr{P}_i + \sum_{j\in\mathcal{S}} \Gamma_i\mathcal{O}_i\mathcal{D}_i k(\mathcal{Q}_j^{\text{MEC}})^2\right)$. It is inherent to perceive that the storage and processing requirement of a service request should not surpass the maximum capability of the selected edge server $\mathcal{S}_j$ and is defined as follows.

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{S}} \Gamma_i\mathcal{O}_i\mathcal{D}_i \le \mathcal{Q}_j^{\text{MEC}}, \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{S} \tag{7}$$

Moreover, processing delay on the IIoT device $\mathcal{M}_i$ and edge server $\mathcal{S}_j$ for each service request must be confined by maximum tolerable delay $\mathcal{T}_i^{\max}$, can be expressed as follows.

$$\sum_{i\in\mathcal{M}} \left\{\mathbf{1}(\Gamma_i = 0)\mathbb{T}_i^{\text{IIoT}} + \mathbf{1}(\Gamma_i \neq 0)\mathbb{T}_i^{\text{MEC}}\right\} \le \mathcal{T}_i^{\max} \tag{8}$$

where, $\mathbf{1}(.)$ is a indicator function and $\mathbf{1}(.) = 1$ once the event is true, otherwise 0. In this model, we exclude the energy consumption and delay overhead of downloading results from the edge server because the downloading data size is as tiny as 1/30 times the original uploading data size. Moreover, the downloading energy consumption is 5.5 times lower than the uploading energy [23]. As a consequence, we ignore the calculation of energy consumption for downloading the result.

C. Problem Formulation

Based on the above derivations, we formulate a service deployment optimization model in which a set of tasks are performed on IIoT devices $\mathcal{M}$, and the remaining tasks are transferred to the edge servers $\mathcal{S}$ for execution. Given the service deployment vector for devices $\Gamma = \{\Gamma_1, \Gamma_2, \ldots \Gamma_I\}$ and resource provisioning vector for edge server $\mathcal{Q} = \{\mathcal{Q}_1^{\text{MEC}}, \mathcal{Q}_2^{\text{MEC}}, \ldots, \mathcal{Q}_J^{\text{MEC}}\}$, we can derive the overall latency $\mathbb{T}^{\text{total}}(\Gamma, \mathcal{Q})$ and energy decay $\mathbb{E}^{\text{total}}(\Gamma, \mathcal{Q})$ for various computing devices as follows.

$$\mathbb{T}_i^{\text{total}}(\Gamma, \mathcal{Q}) = \mathbf{1}(\Gamma_i = 0)\mathbb{T}_i^{\text{IIoT}} + \mathbf{1}(\Gamma_i \neq 0)\mathbb{T}_i^{\text{MEC}}(\Gamma, \mathcal{Q}) \tag{9}$$

$$\mathbb{E}_i^{\text{total}}(\Gamma, \mathcal{Q}) = \mathbf{1}(\Gamma_i = 0)\mathbb{E}_i^{\text{IIoT}} + \mathbf{1}(\Gamma_i \neq 0)\mathbb{E}_i^{\text{MEC}}(\Gamma, \mathcal{Q}) \tag{10}$$

The key objective of this paper is to optimize the weighted sum of energy $\mathbb{E}_i^{\text{total}}(\Gamma, \mathcal{Q})$ and delay $\mathbb{T}_i^{\text{total}}(\Gamma, \mathcal{Q})$ of all IIoT devices $\mathcal{M}$ in the edge-enabled industrial networks, which is represented as $\mathcal{J}(\Gamma, \mathcal{Q}) = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{S}}\big(\alpha\mathbb{E}_i^{\text{total}}(\Gamma, \mathcal{Q}) + \beta\mathbb{T}_i^{\text{total}}(\Gamma, \mathcal{Q})\big)$, where $\alpha+\beta = 1$ and $\alpha, \beta \in [0, 1]$, respectively. Further, the resource constraint $\sum_{i\in\mathcal{M}} \mathbf{1}(\Gamma_i \neq 0)\mathcal{Q}_i \le \mathcal{Q}^{\max}$ is a hard deadline constraint. In addition, there could be no feasible solution that satisfies both $\sum_{i\in\mathcal{M}} \mathbf{1}(\Gamma_i \neq 0)\mathcal{Q}_i \le \mathcal{Q}^{\max}$ and $\sum_{i\in\mathcal{M}} \mathbb{T}_i^{\text{total}}(\Gamma, \mathcal{Q}) \le \mathcal{T}_i^{\max}$. Thus we consider $\mathcal{T}_i^{\max}$ as the soft deadline constraint as presented in [24] and added a penalty function $\sum_{i\in\mathcal{M}} \max[\mathbb{T}_i^{\text{total}}(\Gamma, \mathcal{Q}) - \mathcal{T}_i^{\max}, 0]$ with $\mathcal{J}(\Gamma, \mathcal{Q})$ while making the objective as follows.

$$\begin{aligned}\mathcal{J}(\Gamma, \mathcal{Q}) = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{S}} \Big(\alpha\mathbb{E}_i^{\text{total}}(\Gamma, \mathcal{Q}) + \beta\mathbb{T}_i^{\text{total}}(\Gamma, \mathcal{Q})\Big) \\ + \nu \sum_{i\in\mathcal{M}} \max\Big[\mathbb{T}_i^{\text{total}}(\Gamma, \mathcal{Q}) - \mathcal{T}_i^{\max}, 0\Big]\end{aligned} \tag{11}$$

where ν is a constant positive weight for the penalty function. Thus, the formulated service deployment strategy and the associated QoS constraints for the edge-enabled industrial networks are expressed as follows.

$$\begin{aligned}
&\underset{\Gamma, \mathcal{Q}}{\text{minimize}} && \mathcal{J}(\Gamma, \mathcal{Q}) && \text{(12a)}\\
&\text{subject to} && \Gamma_i \in \{0\} \cup \{1\}, \ \forall i \in \mathcal{M}, && \text{(12b)}\\
& && \mathcal{W}_i \ge 0, \ \forall i \in \mathcal{M}, && \text{(12c)}\\
& && \sum_{i\in\mathcal{M}} \mathbf{1}(\Gamma_i \neq 0)\mathcal{Q}_i \le \mathcal{Q}^{\max}, && \text{(12d)}\\
& && \sum_{i\in|\mathcal{M}|} \Gamma_i = 1, \ \forall i \in \mathcal{M}, && \text{(12e)}\\
& && \sum_{i\in|\mathcal{M}|}\sum_{j\in|\mathcal{S}|} \Gamma_i \le |\mathcal{S}|, \ \forall j \in \mathcal{S}, && \text{(12f)}\\
& && \mathcal{Q}_i^{\text{IIoT}} \ge 0, \text{ and } \mathcal{Q}_j^{\text{MEC}} \ge 0, && \text{(12g)}
\end{aligned}$$

where $\Gamma = \{\Gamma_i | \ \forall i \in \mathcal{M}\}$ and $\mathcal{Q} = \big\{\mathcal{Q}_j^{\text{MEC}} | \ \forall j \in \mathcal{S}\big\}$. In the above formulation, constraint (12b) represents the services generated from IIoT devices $\mathcal{M}$ are executed on its own device or request to the edge servers $\mathcal{S}$. The available transmission bandwidth $\mathcal{W}_i$ of IIoT device $\mathcal{M}_i$ is either 0 or positive, as given in constraint (12c). Constraint (12d) specifies that the maximum computation resource $\mathcal{Q}_i^{\max}$ on the edge server cannot exceed the total required computation resources $\sum_{i\in\mathcal{M}} \mathbf{1}(\Gamma_i \neq 0)\mathcal{Q}_i$. Constraint (12e) restricts the service deployment decision $\sum_{i\in\mathcal{M}} \Gamma_i$ to each IIoT device $\mathcal{M}_i$ is at most 1. Constraint (12f) presents the total number of service requests are within the capacity of the edge server $\mathcal{S}_j$. Finally, constraint (12g) defines the non-negative feature of computation resources.

IV. Proposed Strategy

In this work, we intend to collectively optimize the distribution of communication and computation resources in various computing devices to achieve the least possible delay and energy consumption rate for the next-generation of industrial networks. To confirm this, we split the intended service deployment strategy into two phases: at first, a heuristic-based task execution strategy is employed to identify IIoT executable tasks, then we outline a DRL-based service deployment strategy for optimizing delay and corresponding energy consumption over the edge-enabled industrial environment. The detailed study of these strategies are explained below.

A. *Task Execution Strategy*

Execution of each IIoT-generated task is the preprocessing step of the proposed service deployment technique. Initially, the IIoT device $\mathcal{M}_i$ stores all the generated tasks in a local queue for making task execution decisions Γ_i based on two QoS parameters. Denote $x-type$ be the executable tasks *i.e.,* $x-type \in \{i-type, s-type\}$. Here, $i-type$ represents the IIoT executable tasks, and $s-type$ defines server executable tasks. The task execution strategy takes decisions based on data input size $\mathcal{O}_i$ and CPU clock speed $\mathcal{D}_i$. This strategy also consider CPU-bound $\mathcal{D}^{\max}$ and memory-bound $\mathcal{O}^{\max}$ (IIoT decided parameter) to restrict the computation overhead in IIoT devices $\mathcal{M}$. Thus we have the following policies for each device $\mathcal{M}_i$ at the initial stages of time t.

- $i-type$ tasks: IIoT device $\mathcal{M}_i$ creates a task list for all the executable tasks, where each task must satisfy $i-type = \big\{\mathcal{M}_i \in \mathcal{M} \mid (\mathcal{O}_i \leq \mathcal{O}^{\max}) \cap (\mathcal{D}_i \leq \mathcal{D}^{\max})\big\}$, $\forall i \in \mathcal{M}$. These $i-type$ tasks are finally processed in the IIoT device $\mathcal{M}_i$.
- $s-type$ tasks: Similarly unsatisfied tasks are kept in a separate list, where each task set must satisfy $s-type = \big\{\mathcal{M}_i \in \mathcal{M} \mid (\mathcal{O}_i, \mathcal{D}_i \geq \mathcal{O}^{\max}, \mathcal{D}^{\max}) \cup (\mathcal{O}_i \geq \mathcal{O}^{\max}) \cup (\mathcal{D}_i \geq \mathcal{D}^{\max})\big\}, \forall i \in \mathcal{M}$. Due to the limited computational capacity $\mathcal{Q}_i^{\text{IIoT}}$ of IIoT device $\mathcal{M}_i$, $s-type$ tasks are not executed on IIoT device $\mathcal{M}_i$ and request additional assistance from the edge server $\mathcal{S}_j$.

From the above execution policy, we can define a symmetric difference relation between $i-type$ and $s-type$ with respect to task execution as $i-type\ \Delta\ s-type = \big\{\mathcal{M}_i \mid (\mathcal{M}_i \in i-type \cup \mathcal{M}_i \in s-type) \cap \mathcal{M}_i \notin (i-type \cap s-type)\big\}$ *i.e.,* each task strictly follow these relations for execution. The detailed steps are summarized in *Algorithm* 1.

Algorithm 1: *Task Execution* algorithm

```
1 INPUT: $\mathcal{M}$, $\mathcal{O}_i$ and $\mathcal{D}_i$
2 OUTPUT: Task execution decision
  1: Initialize $x-type$ tasks with attributes $\mathcal{O}_i$ and $\mathcal{D}_i$
  2: Initialize $\mathcal{D}^{\max}$ and $\mathcal{O}^{\max}$
  3: for $i$ = 1 to $|\mathcal{M}|$ do
  4:    if $(\mathcal{O}_i \leq \mathcal{O}^{\max}) \cap (\mathcal{D}_i \leq \mathcal{D}^{\max})$ then
  5:       Execute tasks on IIoT device $\mathcal{M}_i$
  6:    end if
  7:    if $(\mathcal{O}_i, \mathcal{D}_i \geq \mathcal{O}^{\max}, \mathcal{D}^{\max}) \cup (\mathcal{O}_i \geq \mathcal{O}^{\max})$
        $\cup (\mathcal{D}_i \geq \mathcal{D}^{\max})$ then
  8:       Request for additional services from server $\mathcal{S}_j$
  9:    end if
 10: end for
```

B. *DRL-Based Service Deployment*

The key objective of adopting the DRL technique is to intelligently optimize the devices dynamic service requests and minimize the utilization of energy consumption rates $\mathbb{E}^{\text{total}}$ on edge servers $\mathcal{S}$. DRL combines the Deep Neural Network (DNN) and RL into the solution, enabling agents to obtain the best decision from unorganized sets of inputs. Moreover, this dynamic nature of the environment can be expressed as a Markov Decision Process (MDP) based state-action-reward problem. MDP helps in solving a problem when probability or rewards are untold. Moreover, it provides a viable solution that will eventually converge to an optimal solution.

1) MDP-based Learning: The problem of intelligent service deployment strategy for the IIoT devices $\mathcal{M}$ can be formulated as a finite MDP with four attributes $\mathsf{X} = (\mathsf{S}, \mathsf{A}, \mathsf{R}, \mathsf{P})$, where S signifies the system state, A denotes the action space and $\mathsf{P}(s_{t+1}|s_t, a_t)$ denotes the transition probability between state $s_t \in \mathsf{S}$ and $s_{t+1} \in \mathsf{S}$, and $\mathsf{R}(s_t, a_t)$ represents the immediate reward [25]. In general, an agent makes a transition from state s_t to s_{t+1} using action $a \in \mathsf{A}$ and obtain immediate reward $\mathsf{R}(s_t, a_t)$ with probability $\mathsf{P}(s_{t+1}|s_t, a_t)$, which maps state to action defined by a policy π, i.e., $\pi(s_t) = a_t$. The primary goal of MDP is to get optimal policy $\pi^*(s_t)$ busing the best possible action a^* in order to maximize reward R over the long run. Therefore, to define MDP, we begin by describing the essential factors of DRL, namely the state, action, and reward.

• **State :** To evaluate the service deployment strategy as an MDP problem, we define system states for reaching accurate service deployment decisions at the time of t as below.

$$\begin{aligned} \mathsf{S} &= \{s_t = (\Gamma(t),\ \mathcal{Q}(t)\} \\ &= \{\Gamma_1(t), \Gamma_2(t), \ldots, \Gamma_I(t), \mathcal{Q}_1(t), \ldots, \mathcal{Q}_J(t)\} \end{aligned} \tag{13}$$

Specifically, the system state s_t is a $1 \times (IJ)$ dimensional vector, which incorporates all the devices service deployment decisions $\big(\Gamma_i(t) \in \{0\} \cup \{1\}\big)_{\forall i \in \mathcal{M}}$ and corresponding computational resources $\big(\mathcal{Q}_j(t) \in [0, \mathcal{Q}^{\max}]\big)_{\forall j \in \mathcal{S}}$ in the environment.

• **Action :** For the DRL technique associating with the IIoT device $\mathcal{M}_i$, the action $a_t \in \mathsf{A}$ represents the desired service deployment decisions on edge server $\mathcal{S}_j$, defined as follows.

$$\mathsf{A} = \{a_t = \{a_1(t), a_2(t), \ldots, a_I(t)\} \mid a_i(t) \in a^{\max}\} \tag{14}$$

where $a^{\max}$ denotes the maximum number of service requests, processed by the edge server $\mathcal{S}_j$. In practice, MDP converges fastly to an optimal policy $\pi^*(s_t)$ by taking appropriate actions A over the industrial environment.

• **Reward :** Given a particular state s_t and action a_t, the industrial environment makes the error-free decision to reach s_{t+1} from s_t, where the primary goal is to maximize long term reward $\mathsf{R}(.)$ and minimize delay-energy rate $\mathcal{J}(.)$ of each device. For notational simplicity, we use $\mathcal{J}_{s_t}(\Gamma, \mathcal{Q}) = \mathcal{J}(\Gamma(t), \mathcal{Q}(t)\}$ as the objective function. Therefore, we define the immediate reward $\mathsf{R}(s_t, a_t)$ for the state-action pair (s_t, a_t) as follows.

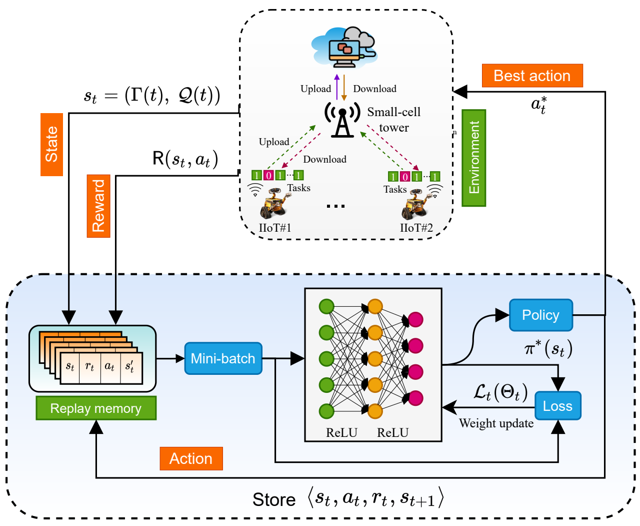


Fig. 2. Illustration of DRL-based service deployment strategy.

$$\mathsf{R}(s_t, a_t) = \mathscr{J}_{s_t}(\Gamma, \mathcal{Q}) - \mathscr{J}_{s_{t+1}}(\Gamma, \mathcal{Q}) \tag{15}$$

2) Proposed DRL-based Solution: To find the solution to the above MDP-based problem, we introduce the well-known Deep Reinforcement Learning (DRL) algorithm as illustrated in Fig. 2. DRL is an online model-free learning approach, which provides a viable solution in larger state-action (s_t, a_t) space, where an agent (IIoT device $\mathcal{M}_i$) interacts with the environment and selects action a_t through policy π for state s_t in a fashion that maximizes the expected long-term reward in discrete timestamp t [26], which is given by.

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\Big[r_t + \sum_{b=1}^{\infty} \gamma^b r_{t+b} | s_t = s, a_t = a, \pi\Big] \tag{16}$$

where $\mathbb{E}[.]$ represents the expectation of a function, and γ is the discount factor. Furthermore, the system uses a gradient descent algorithm to update loss and is represented as follows.

$$\mathcal{L}_t(\Theta_t) = \mathbb{E}_{s_t,a_t,r_t,s_{t+1}\sim \mathsf{P}(.)}\Big[\big(\mathcal{Y}_t - Q(s_t,a_t;\Theta_t)\big)^2\Big] \tag{17}$$

where Θ_t denotes the weight vector of the neural network. Further, $\mathcal{Y}_t$ is called the TD (temporal difference) target and can be expressed as follows.

$$\mathcal{Y}_t = r_t + \gamma \max_{a'} Q\big(s_{t+1}, a'; \Theta_{t-1}\big) \tag{18}$$

Here, weights Θ are used to minimize the squared error between predicted $Q(s_t, a_t)$ and targeted $r_t + \gamma \max_{a'} Q(s_{t+1}, a')$ value. We adopt the experience replay memory ζ technique to reduce the complexity of larger state-action space. The idea is to store the recent state-action pairs (s_t, a_t) in a large memory buffer as experiences. Once the memory is full, less frequently used samples are replaced with current or frequently used examples. Hence search space is also reduced to the size of the replay memory ζ. For faster convergence, adaptive moment estimation (`Adam`) is used to train the network parameters.

3) Learning Process: The learning mechanism for DRL appears to be different from that of a traditional deep learning model, where a batch of training samples are fed into the target and learning network [17]. Then loss $\mathcal{L}_t(\Theta_t)$ is calculated in the forward propagation by the difference between reward values and weights Θ are adjusted with partial derivative ∇_{Θ_t} of the loss function, which can be derived as follows.

$$\Theta \leftarrow \text{Adam}\ \big(\Theta, \alpha\nabla_{\Theta_t}\mathcal{L}_t(\Theta_t)\big) \tag{19}$$

Further, we can derive the loss function as.

$$\begin{aligned}\mathcal{L}_t(\Theta_t) = &\mathbb{E}_{s_t,a_t,r_t}\Big[\mathbb{V}_s\left(\mathcal{Y}_t\right)\Big] + \\ &\mathbb{E}_{s_t,a_t,r_t,s_{t+1}}\Big[\big(\mathcal{Y}_t - Q(s_t,a_t;\Theta_t)\big)^2\Big]\end{aligned} \tag{20}$$

where $\mathbb{V}_s\left(\mathcal{Y}_t\right)$ defines the value function. From the $\mathcal{L}_t(\Theta_t)$ we can derive the derivative of the loss function as follows.

$$\begin{aligned}\nabla_{\Theta_t}\mathcal{L}_t(\Theta_t) = &\nabla_{\Theta_t}\mathbb{E}_{s_t,a_t,r_t}\Big[\mathbb{V}_s\big(\mathcal{Y}_t\big)\Big] + \\ &\nabla_{\Theta_t}\mathbb{E}_{s_t,a_t,r_t,s_{t+1}}\Big[\big(\mathcal{Y}_t - Q(s_t,a_t;\Theta_t)\big)^2\Big] \\ = &\nabla_{\Theta_t}\mathbb{E}_{s_t,a_t,r_t,s_{t+1}}\Big[\big(\mathcal{Y}_t - Q(s_t,a_t;\Theta_t)\big)^2\Big]\end{aligned} \tag{21}$$

Since $\mathbb{E}_{s_t,a_t,r_t}\big[\mathbb{V}_s\left(\mathcal{Y}_t\right)\big]$ is independent from weight Θ, we simply ignore this term from the partial derivative. Now by substituting the value of $\mathcal{Y}_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \Theta_{t-1})$ in (21), we can derive the gradient of $\mathcal{L}_t(\Theta_t)$ as follows.

$$\begin{aligned}\nabla_{\Theta_t}\mathcal{L}_t(\Theta_t) = &\mathbb{E}_{s_t,a_t,r_t,s_{t+1}}\Big[\big(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \\ &\Theta_{t-1}) - \ Q(s_t,a_t;\Theta_t)\big)\ \nabla_{\Theta_t}Q\big(s_t,a_t;\Theta_t\big)\Big]\end{aligned} \tag{22}$$

Finally weights are updated in the DRL environment as $\Theta = \Theta + \alpha\nabla_{\Theta_t}\mathcal{L}_t(\Theta_t)$. The detailed steps of the proposed DQN-based solution are presented in *Algorithm* 2.

Theorem 1. *For a distributed industrial edge networks with two critical time utilities ($\mathscr{T}_{i1}, \mathscr{T}_{i2}$) and a speedup factor $\mathscr{F}$, the task offloading decision always follows the requirements $\mathbb{T}_i^{IIoT} > \max(\mathscr{T}_{i1}, \mathscr{T}_{i2})$ and $\frac{\mathscr{T}_{i1}}{\mathscr{T}_{i2}} < 1$. Where $\mathscr{T}_{i1}$ and $\mathscr{T}_{i2}$ are the coefficients of processing delay and energy consumption.*

Proof: Let $\mathbb{T}_i^{\text{upload}}$ denote the data transmission time to a edge server. In order to improve system performance, the computation time on remote computing device including transmission and processing should be less than the computation time on the IIoT devices as follows $\mathbb{T}_i^{\text{IIoT}} > \mathbb{T}_i^{\text{process}} + \mathbb{T}_i^{\text{upload}}$. Therefore, it is worth offloading when processing on the remote computing device is faster rather IIoT devices. Similarly to save energy consumption rate, IIoT devices always offload tasks when, $\mathcal{P}_i^m\mathbb{T}_i^{\text{IIoT}} > \mathsf{P}_i^{\text{idle}}\mathbb{T}_i^{\text{process}} + \mathscr{P}_i\mathbb{T}_i^{\text{upload}}$. Let $\mathbb{T}_i^{\text{IIoT}} = \mathscr{F}\mathbb{T}_i^{\text{process}}$, where $\mathscr{F} > 1$ is the speedup indicator for computing devices. Further to improve performance and save energy, task offloading has to satisfy the following conditions.

$$\mathbb{T}_i^{\text{IIoT}} > \frac{\mathbb{T}_i^{\text{IIoT}}}{\mathscr{F}} + \mathbb{T}_i^{\text{upload}} \tag{23}$$

$$\mathcal{P}_i^m\mathbb{T}_i^{\text{IIoT}} > \mathsf{P}_i^{\text{idle}}\frac{\mathbb{T}_i^{\text{IIoT}}}{\mathscr{F}} + \mathscr{P}_i\mathbb{T}_i^{\text{upload}} \tag{24}$$

Algorithm 2: *Service Deployment* algorithm

```
1 INPUT: $\mathcal{M}$, $\mathcal{S}$, $\mathcal{W}_i$, $\mathcal{O}_i$, $\mathcal{D}_i$, $k$, $\mathscr{P}_i$, and $\mathscr{Y}_{i,j}$
2 OUTPUT: $a_t^*$: Best action, $\pi^*(s_t)$: Optimal decision
  1: Initialize replay memory $\zeta$
  2: Initialize state $s_t$, action $a_t$ and weight $\Theta_t$
  3: for $i = 1$ to $\mathcal{M}$ do
  4:     On state $s_t$ execute action $a_t$
  5:     Recognize nest state $s_{t+1}$ and immediate reward $r_t$
  6:     Calculate objective function $\mathcal{J}(\Gamma, \mathcal{Q})$
  7:     Store current information $(s_t, a_t, r_t, s_{t+1})$ into $\zeta$
  8:     Choose mini-batch samples from memory $\zeta$
  9:     Calculate $\mathcal{Y}_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \Theta_{t-1})$
 10:     Update loss function utilizing $\mathcal{Y}_t$ and $Q(s_t, a_t)$ as
         $\mathcal{L}_t(\Theta_t) = \mathbb{E}_{s_t,a_t,r_t,s_{t+1}\sim \mathrm{P}(.)}\left[(\mathcal{Y}_t - Q(s_t, a_t; \Theta_t))^2\right]$
 11:     Update weights as $\Theta \leftarrow$ Adam $\left(\Theta, \alpha\ \nabla_{\Theta_t}\mathcal{L}_t(\Theta_t)\right)$
         where $\Theta =$
         $\Theta + \alpha\ \mathbb{E}_{s_t,a_t,r_t,s_{t+1}}\Big[\big(r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \Theta_{t-1}) -$
         $Q(s_t, a_t; \Theta_t)\big)\ \nabla_{\Theta_t} Q\big(s_t, a_t; \Theta_t\big)\Big]$
 12:     Update the target network
 13:     Take best action $a_t^*$ and accurate decision $\pi^*(s_t)$
 14: end for
```

The inequalities in (23) and (24) maintains larger $\mathscr{F}$, small $\mathcal{O}_i$ and larger $\mathbb{R}_{i,j}$ values. Let $\mathscr{T}_{i1}$ and $\mathscr{T}_{i2}$ be the two time values and when $\mathbb{T}_i^{\text{IIoT}}$ arrives at the equilibrium point in (23) and (24), then according to Wu *et al.* [27] we have.

$$\mathscr{T}_{i1} = \frac{\mathscr{T}_{i1}}{\mathscr{F}} + \mathbb{T}_i^{\text{upload}} \Rightarrow \mathscr{T}_{i1} = \frac{\mathbb{T}_i^{\text{upload}}}{1 - \frac{1}{\mathscr{F}}} \tag{25}$$

$$\mathcal{P}_i^m \mathscr{T}_{i2} > \mathrm{P}_i^{\text{idle}} \frac{\mathscr{T}_{i2}}{\mathscr{F}} + \mathscr{P}_i \mathbb{T}_i^{\text{upload}} \Rightarrow \mathscr{T}_{i2} = \frac{\mathscr{P}_i \mathbb{T}_i^{\text{upload}}}{\mathcal{P}_i^m - \frac{\mathrm{P}_i^{\text{idle}}}{\mathscr{F}}} \tag{26}$$

Equation (25) and (26) requites $1 - \frac{1}{\mathscr{F}} > 0$ and $\mathscr{F} > \frac{\mathrm{P}_i^{\text{idle}}}{\mathcal{P}_i^m}$. Especially when $\mathrm{P}_i^{\text{idle}} = \mathscr{P}_i$, inequality in (24) reduced to $\mathbb{T}_i^{\text{IIoT}} > \frac{\mathrm{P}_i^{\text{idle}}}{\mathcal{P}_i^m}\left(\frac{\mathbb{T}_i^{\text{IIoT}}}{\mathscr{F}} + \mathbb{T}_i^{\text{upload}}\right)$. Therefore, in order to minimize the processing time, meanwhile to extend battery life, $\mathbb{T}_i^{\text{IIoT}}$ has to meet the bellow requirement $\mathbb{T}_i^{\text{IIoT}} > \max(\mathscr{T}_{i1}, \mathscr{T}_{i2})$. Which fulfil the first condition. Further to compare $\mathscr{T}_{i1}$ and $\mathscr{T}_{i2}$, we consider $\frac{\mathscr{T}_{i1}}{\mathscr{T}_{i2}} = \frac{\mathbb{T}_i^{\text{upload}}}{1-\frac{1}{\mathscr{F}}} \cdot \frac{\mathcal{P}_i^m - \frac{\mathrm{P}_i^{\text{idle}}}{\mathscr{F}}}{\mathscr{P}_i \mathbb{T}_i^{\text{upload}}} < 1$, which also satisfy the second condition and this completes the proof.

C. Complexity Analysis

The overall complexity of our service deployment strategy relies on both task execution strategy and DRL-based strategy. In *Algorithm* 1, IIoT executable tasks are identified in $\mathscr{O}(1)$ times. For $|\mathcal{M}|$ number of devices, the complexity will be $|\mathcal{M}| \times \mathscr{O}(1)$, i.e., $\mathscr{O}(|\mathcal{M}|)$, which is polynomial in time. Next, in *Algorithm* 2, the MDP model considers input state as offloading decision Γ_i and computation capability of the edge server $\mathcal{Q}_j^{\text{MEC}}$. Hence, the DRL has $(1 \times IJ)$ dimensional vector as input. Let DNN contain K numbers of neurons in the hidden layer and E numbers of neurons in the output layer. If the DNN contains L hidden layer, then the required multiplication operation is $(1 \times IJ) \times K + L \times K + K \times E = \mathscr{O}(K((1 \times IJ) + L + E))$ and can be simplified into $\mathscr{O}(K(IJ + L + E))$. To be more precise, the complexity of applying the activation function is $\mathscr{O}(L \times K)$. Now the complexity becomes $\mathscr{O}(K(IJ + L + E + L))$ which can be further simplified into $\mathscr{O}(K(IJ + E + L))$. Since the service deployment decision is with $\mathscr{O}(I)$ times. The overall run-time computational complexity becomes $\mathscr{O}(|\mathcal{M}|) + \mathscr{O}(K(IJ + E + L)) \approx \mathscr{O}(K(IJ + F + L))$ as $|\mathcal{M}| << K(IJ + E + L)$.

V. Performance Evaluation

In this section, we examine the performance of the intended DRL-based service deployment strategy with numerous performance metrics. For the comparative analysis, we study four standard service deployment strategies, such as local execution, cloud execution, heuristic `DPTO` [21] and stochastic `EETO` [13] algorithms, respectively.

- **Local execution**: In the local execution strategy, requested services are processed in the IIoT devices without considering the provision of the cloud server. Though this strategy eliminates service deployment process but increases computation overhead among the IIoT devices.
- **Cloud execution**: In the cloud execution strategy, all the requested services are carried to randomly selected computing devices without considering the availability of the storage and computational resources.

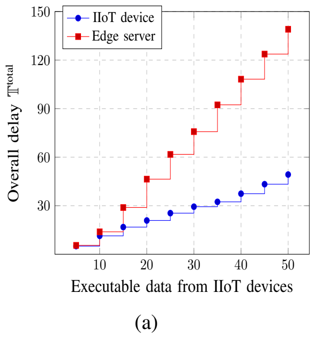
TABLE III
SIMULATION PARAMETERS

Parameters	Values
Number of IIoT devices ($\mathcal{M}$)	[20,50]
Number of edge-enabled small-cell towers ($\mathcal{S}$)	[4,7]
Transmission bandwidth of IIoT device ($\mathcal{W}_i$)	20MHz
Transmission noise for IIoT device ($\mathscr{D}_i$)	10^{-9}W
Service deployment vector of IIoT device (Γ_i)	{0,1}
CPU requirement of the service request ($\mathcal{D}_i$)	1900
CPU frequency of the IIoT device ($\mathcal{Q}_i^{\text{IIoT}}$)	4×10^7
Size of the replay memory (ζ)	50000
Input data size ($\mathcal{O}_i$)	[5,50] Mb
CPU frequency of the remote server ($\mathcal{Q}_j^{\text{MEC}}$)	40×10^9
Task arrival rate on IIoT device (λ_i)	0.5
Transmission power of the IIoT device ($\mathscr{P}_i$)	500mW

A. Simulation Setup

For the experimental analysis, we use an Intel i7 CPU with 10GB RAM and Python language for training this network. More precisely, we use TensorFlow, Keras-rl2 and GYM OpenAI open-source libraries, which are widely adopted for executing DRL-based techniques. In the simulation, we consider a ST base station with radius 60m. We set number of IIoT devices $\mathcal{M} = [20, 50]$, ν = [1.5,2.5], small-cell towers $\mathcal{S} = [4, 7]$, transmission bandwidth $\mathcal{W}_i = 20$MHz, data size $\mathcal{O}_i = [5, 50]$ Mb, noise $\mathscr{D}_i = 10^{-9}$W, and transmission power $\mathscr{P}_i = 500$mw. Further, we consider processing requirement $\mathcal{D}_i = 1900$ [cycles/byte], processing capacity of IIoT device $\mathcal{Q}_i^{\text{IIoT}} = 4 \times 10^7$ and ST server $\mathcal{Q}_j^{\text{MEC}} = 40 \times 10^9$, respectively. Moreover, we consider learning rate 0.001, number of episode 100000, discount factor $\gamma = 0.8$ and batch size {16,32,64}.

Other parameters are considered from [12], [28] and [21], respectively. Detailed simulation parameters are presented in Table III.



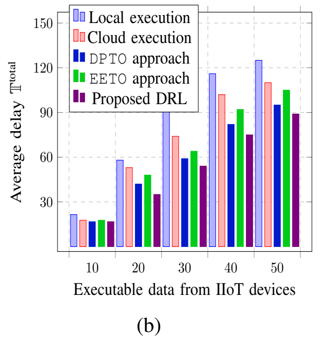


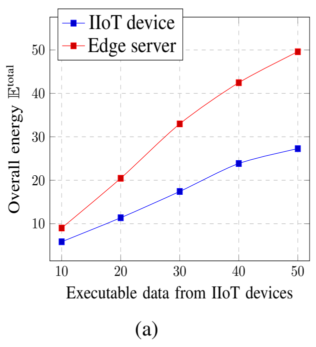
Fig. 3. Delay performance analysis, (a) On various computing devices. (b) Comparison with standard baseline algorithms.

B. Average Delay

The Average delay $\mathbb{T}^{\text{total}}$ of a set of tasks represents the combination of average delay on the IIoT devices $\mathcal{M}$ and edge servers $\mathcal{S}$, where the edge server delay $\mathbb{T}^{\text{MEC}}$ included transmission delay $\mathbb{T}_i^{\text{upload}}$ and processing delay $\mathbb{T}_i^{\text{process}}$ in the small-cell edge networks. In other words, this metric tells us how long the tasks generated through IIoT devices $\mathcal{M}$ had waited before they started processing. From $\mathbb{T}_i^{\text{MEC}} = \sum_{i\in\mathcal{M},j\in\mathcal{S}}\left(\frac{\Gamma_i\mathcal{O}_i}{\mathbb{R}_{i,j}} + \frac{\Gamma_i\mathcal{O}_i\mathcal{D}_i}{\mathcal{Q}_j^{\text{MEC}}}\right)$ it can be observed that the computation delay $\mathbb{T}_i^{\text{MEC}}$ on remote computing device $\mathcal{S}_j$ can be optimized by increasing the data transmission rate $\mathbb{R}_{i,j}$ and CPU frequency $\mathcal{Q}_j^{\text{MEC}}$ of the edge server $\mathcal{S}_j$. Further from $\mathbb{T}_i^{\text{IIoT}} = (1 - \sum_{i\in\mathcal{M}}\Gamma_i)\mathcal{O}_i\mathcal{D}_i/\mathcal{Q}_i^{\text{IIoT}}$ it is also noted that the processing delay $\mathbb{T}_i^{\text{IIoT}}$ on the IIoT device $\mathcal{M}_i$ can be minimized by increasing the CPU frequency on the IIoT device $\mathcal{Q}_i^{\text{IIoT}}$. Fig. 3(a) represents the normalized delay $\mathbb{T}^{\text{total}}$ performance of the network on various computing devices, whereas Fig. 3(b) illustrates the comparative analysis of the proposed DRL-based strategy with standard baseline algorithms. Results show that the proposed strategy performs better as compared with local execution, cloud execution, `DPTO`, and `EETO` strategies while optimizing average delay upto 23%, 21%, 16%, and 18%, respectively. Specifically, the task execution strategy assists in reducing the computation overload from the IIoT devices while overcoming the complexity of the decision-making process. Thus, overall computation time also reduces from edge-enabled industrial networks.

C. Average Energy Consumption

The overall energy consumption $\mathbb{E}^{\text{total}}$ for a set of service requests can be determined by processing energy $\mathbb{E}_i^{\text{IIoT}}$ on the local IIoT device $\mathcal{M}_i$, transmission energy consumption $\mathbb{E}_i^{\text{upload}}$ on the IIoT device, and processing energy consumption $\mathbb{E}_i^{\text{process}}$ on the remote computing device. From the formulation $\mathbb{E}_i^{\text{IIoT}} = (1 - \sum_{i\in\mathcal{M}}\Gamma_i)\mathcal{O}_i\mathcal{D}_i k\left(\mathcal{Q}_i^{\text{IIoT}}\right)^2$ and $\mathbb{E}_i^{\text{MEC}} = \sum_{i\in\mathcal{M},j\in\mathcal{S}}\left(\frac{\Gamma_i\mathcal{O}_i}{\mathbb{R}_{i,j}}\mathscr{P}_i + \sum_{j\in\mathcal{S}}\Gamma_i\mathcal{O}_i\mathcal{D}_i k\left(\mathcal{Q}_j^{\text{MEC}}\right)^2\right)$, the energy consumption depends on the data size $\mathcal{O}_i$ and CPU frequency of the IIoT device $\mathcal{Q}_i^{\text{IIoT}}$. Whereas the energy consumption on remote computing device $\mathbb{E}_i^{\text{MEC}}$ is directly depends on transmission power $\mathscr{P}_i$ of the IIoT device and the CPU frequency $\mathcal{Q}_j^{\text{MEC}}$ of the computing server $\mathcal{S}_j$. Fig. 4(a) represents the amount of normalized energy consumption $\mathbb{E}^{\text{total}}$ rate on various computing devices while executing a set of service requests and Fig. 4(b) depicts the energy-based performance analysis of our DRL-base strategy with existing algorithms. It is interesting to notice that the proposed strategy reduces the energy consumption rate $\mathbb{E}^{\text{total}}$ upto 13%, 11%, 8%, and 9% as compared with local execution, cloud execution, `DPTO`, and `EETO` strategies while satisfying the maximum number of service requests within the delay bound $\mathcal{T}^{\max}$. The reason is that the proposed DRL-based strategy considers the offloading variable $\Gamma(t)$ and currently available resources $\mathcal{Q}(t)$ of various computing devices for making the accurate decision $\pi^*(s_t)$ over the industrial networks.



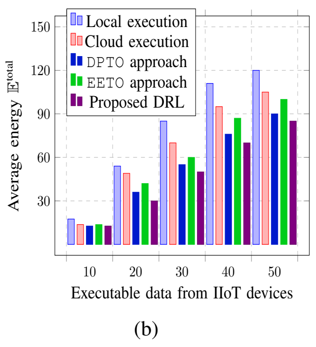
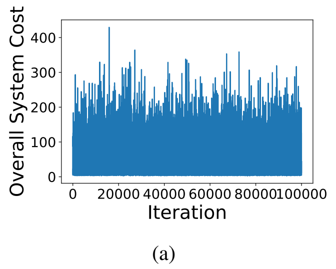


Fig. 4. Energy performance analysis, (a) On various computing devices. (b) Comparison with standard baseline algorithms.



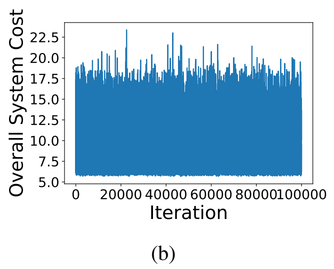


Fig. 5. Performance analysis of the industrial edge networks, (a) Using random offloading strategy. (b) Using MDP based state-action-reward policy.

D. Training of DRL Technique

This metric represents the run-time performance of the industrial environment, where the implementation of the DRL strategy largely depends on the number of hidden layers, an output layer, and input data size $\mathcal{O}_i$. For training the DRL model, we define a `MobileEnv` environment with four functions as `Init`, `CalReward`, `Step`, and `Reset`. At first, system performance $\mathcal{J}(\Gamma,\mathcal{Q})$ is considered using random offloading strategy presented in Fig. 5(a). Then we study the MDP-based state-action-reward approach and calculate objective $\mathcal{J}(\Gamma,\mathcal{Q})$ with random values. Surprisingly, MDP reduces overall system performance up to 10-20 times, as

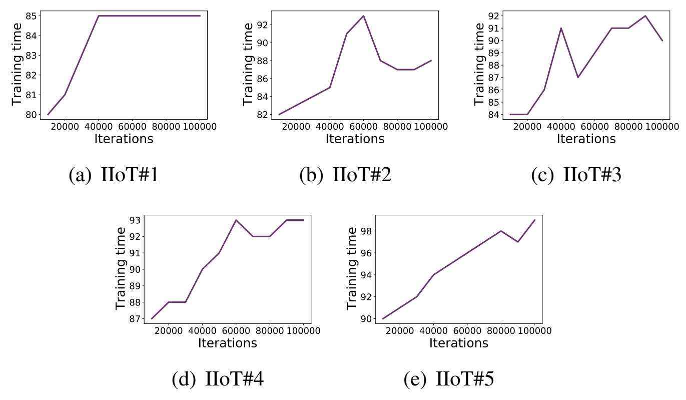


Fig. 6. Run-time complexity using DRL technique on, (a) IIoT device#1. (b) IIoT device#2. (c) IIoT device#3. (d) IIoT device#4 and (e) IIoT device#5.

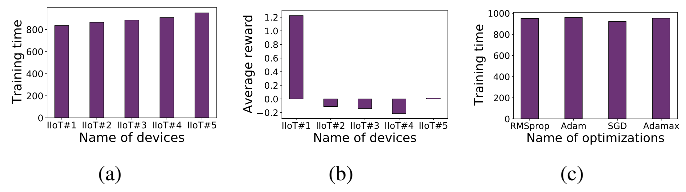


Fig. 7. Training performance, (a) Various IIoT devices for execution time. (b) Average reward on IIoT devices. (c) With various optimization techniques.

illustrated in Fig. 5(b). Further, to optimize the edge resources $\mathcal{J}(\Gamma, \mathcal{Q})$, we define a DNN network with 3 hidden layers and 1 output layer. We import the hidden layers as dance layer, activation as ReLU, and for the last layer, we took the Softmax function. To iterate the DRL over the entire environment, we run the model up to 100000 independent iterations with 4 optimization techniques and plot the run-time complexity for five different devices in Fig. 6 and Fig. 7. Moreover, for four different optimizer, such as RMSprop, Adam, SGD, and Adamax, the computational complexity are 950.693, 960.727, 922.164, and 953.491 (in second) time, which is also depicted in Fig. 7(c). On the other hand, the overall system performance $\mathcal{J}(\Gamma, \mathcal{Q})$, as shown in Fig. 8(a) also optimized over the industrial edge networks, and the proposed DRL technique obtained better results compared with standard algorithms. Similar types of analysis are also found with varying the computational capacity $\mathcal{Q}_j^{\text{MEC}}$ of the edge server $\mathcal{S}_j$ in Fig. 8(b).

E. Reward Calculation

The reward function $\mathrm{R}(s_t, a_t)$ allows the DRL technique to conclude a decision Γ_i instead of arriving at a prediction. For obtaining maximum reward from the proposed environment, we train the DRL model up to 100000 iterations, where the best set of hyper-parameters is considered for performance improvement. We consider four popular neural network optimizer such as Adam, RMSprop, SGD, and Adamax for reward calculation. Moreover, The policy is taken as BoltzmannQPolicy, sequential memory limit in 50000, error metric as Mean Absolute Error (MAE), and total

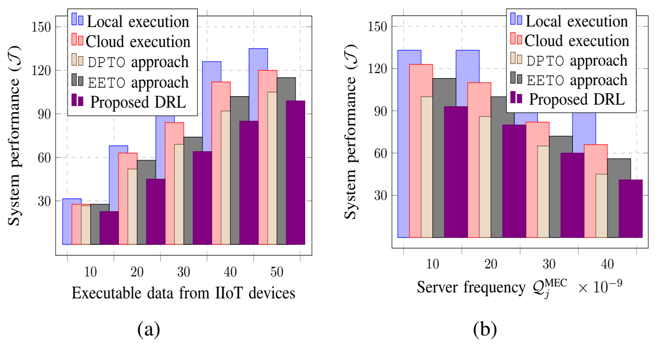


Fig. 8. Overall system performance, (a) Varying the number of IIoT devices $\mathcal{M}$. (b) Varying the computation frequency on the edge server $\mathcal{S}_j$.

trainable parameters are 11,744. Further, we generate training samples through a random process. Fig. 9(a) shows the system reward on the industrial edge environment using random action strategy, and Fig. 9(b) illustrates the overall system reward of the DRL-based method with various networks optimization techniques. From Fig. 9(b) we can observe that the performance of the Adam optimizer is comparatively better than other optimization techniques. Besides that, Adam joins the most valuable features of the RMSProp and AdaGrad algorithms to handle sparse gradients in a noisy environment. The analysis also confirms that the proposed DRL technique obtains high reward values compared with the random service deployment strategy.

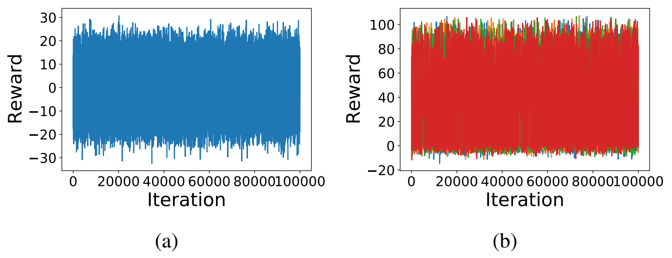


Fig. 9. Performance analysis for calculating reward, (a) Using random action strategy. (b) Using proposed DRL-based strategy.

The preceding experimental analysis confirms that the proposed service deployment technique on the edge environment indeed determines the responsibilities and difficulties of the existing edge server and industrial networks with the DRL technique. It operates efficiently under the massive service load and reduces the overall energy consumption ($\mathbb{E}^{\text{total}}$) and delay ($\mathbb{T}^{\text{total}}$) while maximizing task execution rate.

VI. Conclusion

This paper investigates the service deployment and task execution problem for all the IIoT-generated tasks in the edge-enabled industrial networks intending to optimize the weighted energy and delay. Despite its nature of being a mixed binary nonlinear programming, we first sketch a heuristic-based task execution strategy for choosing IIoT executable tasks and then proposed a DRL-based service deployment framework, which considers both the IIoT service requests and available computing resources in the industrial environment. Moreover, the proposed approach is intelligent in delivering close-to-optimal solutions while achieving several deadline constraints. Extensive simulation on various industrial and network-level parameters guarantees that the proposed strategy has the advantage of utilizing maximum computation resources compared with baseline techniques. As a part of future work, we will extend the proposed service deployment strategy for healthcare applications and optimize objective function for delay-restricted real-time applications.

VII. Acknowledgement

This work is supported by DST (SERB), Government of India, under Grant EEQ/2018/000888 and UoH-IoE by MHRD, India (F11/9/2019-U3(A)).

References

[1] M. J. Piran, S. Verma, V. G. Menon, and D. Y. Suh, "Energy-Efficient Transmission Range Optimization Model for WSN-Based Internet of Things," *Computers, Materials and Continua*, vol. 67, no. 3, pp. 2989–3007, 2021.

[2] X. Xu, B. Shen, X. Yin, M. R. Khosravi, H. Wu, L. Qi, and S. Wan, "Edge Server Quantification and Placement for Offloading Social Media Services in Industrial Cognitive IoV," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2910–2918, 2021.

[3] G. Han, J. Wu, H. Wang, M. Guizani, J. A. Ansere, and W. Zhang, "A Multicharger Cooperative Energy Provision Algorithm Based on Density Clustering in the Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9165–9174, 2019.

[4] P. Goswami, A. Mukherjee, M. Maiti, S. K. S. Tyagi, and L. Yang, "A Neural Network Based Optimal Resource Allocation Method for Secure IIoT Network," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[5] X. Xu, Q. Huang, X. Yin, M. Abbasi, M. R. Khosravi, and L. Qi, "Intelligent Offloading for Collaborative Smart City Services in Edge Computing," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7919–7927, 2020.

[6] 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 of Things Journal*, pp. 1–1, 2020.

[7] Y. Ren, Y. Sun, and M. Peng, "Deep Reinforcement Learning Based Computation Offloading in Fog Enabled Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4978–4987, 2021.

[8] X. Xu, B. Shen, S. Ding, G. Srivastava, M. Bilal, M. R. Khosravi, V. G. Menon, M. A. Jan, and W. Maoli, "Service Offloading with Deep Q-Network for Digital Twinning Empowered Internet of Vehicles in Edge Computing," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[9] P. Goswami, A. Mukherjee, P. Chaterjee, and L. Yang, "An Optimal Resource Allocation Method for IIoT Network," in *Adjunct Proceedings of the 2021 International Conference on Distributed Computing and Networking*, ser. ICDCN '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 31–36. [Online]. Available: https://doi.org/10.1145/3427477.3429988

[10] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep Reinforcement Learning-Based Adaptive Computation Offloading for MEC in Heterogeneous Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7916–7929, 2020.

[11] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas, and S. Khaf, "A Deep Learning Approach for Energy Efficient Computational Offloading in Mobile Edge Computing," *IEEE Access*, vol. 7, pp. 149 623–149 633, 2019.

[12] 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.

[13] A. Hazra, M. Adhikari, T. Amgoth, and S. Srirama, "Joint Computation Offloading and Scheduling Optimization of IoT Applications in Fog Networks," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2020.

[14] S. Iqbal, R. M. Noor, A. W. Malik, and A. U. Rahman, "Blockchain-enabled adaptive learning-based resource sharing framework for IIoT environment," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[15] S. Misra, S. P. Rachuri, P. K. Deb, and A. Mukherjee, "Multi-Armed Bandit-based Decentralized Computation Offloading in Fog-Enabled IoT," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[16] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-Optimal Dynamic Computation Offloading for Industrial IoT in Fog Computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 566–576, 2020.

[17] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and Energy-Efficient Computation Offloading in Mobile Edge Computing Using Deep Reinforcement Learning," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2021.

[18] X. Chen and G. Liu, "Energy-Efficient Task Offloading and Resource Allocation via Deep Reinforcement Learning for Augmented Reality in Mobile Edge Networks," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[19] W. Sun, J. Liu, and Y. Yue, "Ai-Enhanced Offloading in Edge Computing: When Machine Learning Meets Industrial IoT," *IEEE Network*, vol. 33, no. 5, pp. 68–74, 2019.

[20] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent Deep Reinforcement Learning for Joint Multichannel Access and Task Offloading of Mobile-Edge Computing in Industry 4.0," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6201–6213, 2020.

[21] 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 of Things Journal*, vol. 7, no. 7, pp. 5773–5782, 2020.

[22] J. Li, A. Wu, S. Chu, T. Liu, and F. Shu, "Mobile Edge Computing for Task Offloading in Small-Cell Networks via Belief Propagation," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[23] 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 Transactions on Communications*, vol. 68, no. 3, pp. 1524–1537, 2020.

[24] S. Nath, Y. Li, J. Wu, and P. Fan, "Multi-user Multi-channel Computation Offloading and Resource Allocation for Mobile Edge Computing," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[25] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Collaborative AI-enabled Intelligent Partial Service Provisioning in Green Industrial Fog Networks," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[26] D. Van Le and C. Tham, "A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 760–765.

[27] H. Wu, Q. Wang, and K. Wolter, "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems," in *2013 IEEE International Conference on Communications Workshops (ICC)*, 2013, pp. 728–732.

[28] W. Zhang, J. Wang, G. Han, S. Huang, Y. Feng, and L. Shu, "A Data Set Accuracy Weighted Random Forest Algorithm for IoT Fault Detection Based on Edge Computing and Blockchain," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2354–2363, 2021.