Sustainable Edge Node Computing Deployments in Distributed Manufacturing Systems

S. Goudarzi, *Senior Member, IEEE,* S. A. Soleymani, *Member, IEEE,* M. H. Anisi, *Senior Member, IEEE,* A. Jindal, *Senior Member, IEEE,* F. Dinmohammadi, *Senior Member, IEEE,* Pei Xiao, *Senior Member, IEEE*

***Abstract*—The advancement of mobile internet technology has created opportunities for integrating the Industrial Internet of Things (IIoT) and edge computing in smart manufacturing. These sustainable technologies enable intelligent devices to achieve high-performance computing with minimal latency. This paper introduces a novel approach to deploy edge computing nodes in smart manufacturing environments at a low cost. However, the intricate interactions among network sensors, equipment, service levels, and network topologies in smart manufacturing systems pose challenges to node deployment. To address this, the proposed sustainable game theory method identifies the optimal edge computing node for deployment to attain the desired outcome. Additionally, the standard design of Software Defined Network (SDN) in conjunction with edge computing serves as forwarding switches to enhance overall computing services. Simulations demonstrate the effectiveness of this approach in reducing network delay and deployment costs associated with computing resources. Given the significance of sustainability, cost efficiency plays a critical role in establishing resilient edge networks. Our numerical and simulation results validate that the proposed scheme surpasses existing techniques like shortest estimated latency first (SELF), shortest estimated buffer first (SEBF), and random deployment (RD) in minimizing the total cost of deploying edge nodes, network delay, packet loss, and energy consumption.**

***Index Terms*—Game theory, smart manufacturing, edge node selection, SDN.**

I. Introduction

INDUSTRY 5.0 pertains to a smart manufacturing setting where people, technology infrastructure, and sensor-equipped machinery work together to produce and transmit a large volume of data, commonly known as Big Data. The development of networking and communication technologies, artificial intelligence, distributed computing, and beyond 5G is giving rise to a new era known as Industry 5.0. Within the realm of technological progress, Industry 4.0 witnesses a heightened incorporation of robotics in manufacturing. Nevertheless, Industry 5.0 is regarded as a progression in the partnership between humans and automation, accentuating a cooperative and safe alliance where both humans and machines actively cooperate in order to fulfil objectives, rather than machines completely supplanting humans.

Numerous research have delved into the progress and obstacles entailed by Industry 5.0 [1]–[4]. However, these preceding inquiries have either adopted a broad overview or specialized in particular domains. As an illustration, a framework that integrates edge AI was suggested for scrutinizing electricity data originating from smart meters [5]. The results of the experiments demonstrated that deploying the AI engine at the edge not only simplifies system intricacies and reduces processing latencies but also heightens the precision of electric load forecasts. Nonetheless, the limited computational capabilities of edge devices present challenges when conducting learning tasks directly on them. Moreover, establishing secure environments for model learning on edge devices is a complex endeavor. Privacy apprehensions among data proprietors restrict data exchange in edge AI, diverging from cloud-based learning tasks where data sharing is more prevalent within a secure setting. In response to this challenge, an E-Tree learning framework was introduced in [6], employing decentralized model aggregation to empower edge devices in conducting localized and gradual model aggregation. On-device AI, also recognized as embedded AI, empowers data processing and computation at the point of data origination. Given the escalating data generation at IoT endpoint devices, the imperative to maximize insights without necessitating data transit becomes evident. Nonetheless, on-device AI grapples with challenges such as the resource-intensive demands of ML algorithms and the limited longevity of IoT end-device batteries. The adoption of tinyML, an emerging subset of on-device AI, addresses these concerns by enabling ML processing on low-power IoT devices [7]. A distinctive aspect of TinyML is its ability to incorporate pre-trained machine learning models onto the edge, thus providing ML-as-a-Service (MLaaS) to IoT end devices.

Also, the advanced information technologies, including the Industrial Internet of Things (IIoT) [8], [9], cloud computing [10], and digital twin [11], have gained significant traction within the manufacturing industry. As a result, there has been rapid development and widespread implementation [12] of these technologies, leading to various applications and advancements in the sector. This shift from digital to intelligent enterprise manufacturing paradigms has been accelerated by

S. Goudarzi and F. Dinmohammadi are with the School of Computing and Engineering, University of West London, St Mary's Rd, Ealing, London, W5 5RF, UK. (e-mail:shidrokh.goudarzi@uwl.ac.uk; e-mail: fateme.dinmohammadi@uwl.ac.uk).

S. A. Soleymani is with the Centre for Vision Speech and Signal Processing (CVSSP), University of Surrey, Guildford GU2 7XH, U.K. (e-mail: s.soleymani@surrey.ac.uk).

M. H. Anisi is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, United Kingdom (e-mail: m.anisi@essex.ac.uk).

A. Jindal is with the Department of Computer Science, Durham University, Durham DH1 3LE, United Kingdom (e-mail: anish.jindal@durham.ac.uk).

P. Xiao is with the 5G&6G Innovation Centre, Institute for Communication Systems (ICS), University of Surrey, Guildford, U.K. (e-mail: p.xiao@surrey.ac.uk).

the Industry 4.0 initiative, leading to significant advancements in group-distributed manufacturing across various areas such as hardware and production facilities. Smart objects enabled by IoT are integrated with intelligent production processes in the Industry 4.0 context, allowing for real-time communications and collaborations among humans, machines, and sensors to make informed decisions [13]. The IIoT smart objects generate vast amounts of data that contain valuable information, which can be utilized to improve production processes.

Sustainable manufacturing, widely adopted as an economically efficient approach in contemporary manufacturing sectors, empowers manufacturers to execute their development strategies effectively while minimizing environmental impact and optimizing resource utilization throughout the product development lifecycle [14]. Researchers have conducted extensive investigations into the integration of IIoT, cloud computing, automation, and artificial intelligence in smart manufacturing, enhancing both machine and product intelligence [15]. Nevertheless, traditional cloud computing falls short in meeting the real-time demands of production processes. To address this challenge, edge computing is seamlessly integrated into the IIoT, enabling timely execution of specialized tasks within the premises of smart factories or industries [16]. Edge computing encompasses a platform strategically positioned in close proximity to device or data sources, amalgamating computing, storage, networking, and applications to deliver intelligent edge services. These services encompass dynamic connectivity, real-time data processing, data analysis, and privacy protection. In the realm of intelligent manufacturing, where swift detection, control, and execution hold paramount importance, the deployment of edge computing nodes emerges as an indispensable solution to facilitate enhanced operational efficiency.

Numerous studies have examined edge computing and the deployment of edge computing nodes in intelligent manufacturing, with significant findings. For instance, Li et al. [14] suggested a flexible transmission structure that combines software-defined networks (SDN) and edge computing. Their method solves issues related to exchanging data with various delay streams among multiple smart devices in the IIoT system. This architecture integrates global centralized SDN and edge computing. In [17], a method for placing virtual machine clusters in software-defined data centres was presented that takes into account energy efficiency and ensures the quality of service (QoS) and service level agreement (SLA) management. Their approach involves considering the similarities among virtual machines and formulating the deployment process as a weighted directed graph. By placing requests on energy-efficient virtual machines, the algorithm aims to provide better QoS to users.

Mobile edge computing (MEC), as a subset or specific implementation of edge computing, plays a crucial role in supporting numerous IIoT applications by offering advantages such as low latency, high resilience, affordability, robust connectivity, and security for M2M communications within the IIoT domain [14]. The combination of MEC and IIoT has gained significant attention from researchers. For instance, an adaptive transmission architecture utilizing SDN and edge computing was proposed in [18] to enable data exchange among smart devices in IIoT scenarios with varying delay requirements. Another work introduced a smart resources partitioning scheme based on service popularity for IIoT enabled by edge computing [18]. Furthermore, the study [19] presented an SDN-based edge-cloud interplay to handle streaming big data in the edge computing-enabled IIoT environment. Authors in [20] examined the applicability of edge computing as a middleware to various industrial scenarios. They emphasized the capability of edge computing to process unstructured big data locally in the industry premises before transmitting it to the cloud. Similarly, Mike Jia et al. [21] investigated the distribution of users and the placement of a limited number of micro-clouds in metropolitan area networks to reduce task offloading's average waiting time.

However, previous research efforts have not addressed the crucial challenge of deploying edge computing nodes in intelligent manufacturing settings. Deploying these nodes is particularly challenging due to constraints such as limited computing power, equipment heterogeneity, varying service levels, and complex network topologies. These complexities arise from the intricate network interactions among equipment controllers, sensors, and diverse machinery [22]. Consequently, there is a clear need for a robust and real-time data processing approach to effectively deploy edge computing nodes within the context of smart manufacturing [23]. This highlights the fact that the deployment strategies discussed in prior studies are ill-suited for the specific requirements of smart manufacturing. In contrast to earlier research, which primarily focused on energy optimization for mobile devices, it is worth noting that in intelligent manufacturing, the devices are predominantly stationary, and energy consumption is not the primary concern under consideration.

The goal of this study is to use edge computing nodes in smart manufacturing, and it proposes a method to select the best edge nodes for computing tasks while minimizing delay and deployment cost. The edge nodes will act as computing nodes and provide edge computing services to the network nodes. To achieve this, the study suggests using game theory to optimize the selection of edge nodes and ensure effective task computation for all network nodes.

The objective of this research is to improve the performance of data processing on a network node by optimizing the placement of edge computing nodes in edge networks. The main focus is on smart manufacturing in Industry 5.0. To achieve this, the study proposes a game theory model, where the edge nodes compete to win the connection of the source node. The model considers the quality of service requested by the node, including the minimum network delay and computing resources deployment cost. By solving a cooperative game for the edge computing node selection problem, the study aims to identify the best edge node selection policies for intelligent manufacturing, taking into account competition among different edge nodes to ensure effective task processing. The ultimate goal is to improve the efficiency and stability of data processing in smart manufacturing environments.

The key achievements of this research are twofold. Firstly, it examines a method for deploying edge computing nodes

in manufacturing environments. Secondly, it enhances the efficiency and reliability of data processing. To summarize, the main contributions of this study are:

1) We design an SDN-based architecture that is efficient for edge computing in smart manufacturing. This architecture can be used to implement and test their proposed approach to selecting the optimal edge computing nodes.
2) We propose an adaptive model based on game theory, which helps select the most suitable edge node among multiple edge nodes in smart manufacturing during the production process.
3) We create an SDN-edge node architecture that can address the data processing constraints. By leveraging SDN technology, which provides a comprehensive network view, we formulate end-to-end policies with high computational capability. Using this approach, we identify SDN-edge node configurations that improve both network delay and computing resource deployment cost automatically.

The remainder of the paper is structured as follows. In Section II, we present an architectural design that includes all the design requirements of the proposed model. Section III presents the edge computing node selection as an optimization problem and explains the designed game theory model. Then, in Section IV, we describe the proposed model for selecting the best edge node. Next, Section V discusses the simulation results. Finally, Section VI concludes the manuscript with final remarks and future research directions.

II. Network Model

The proposed model requires an architectural design that satisfies specific requirements. Figure 1 illustrates an SDN-edge-based system where each edge node performs task computation for the network nodes. The bottom layer consists of edge devices in Industry 5.0, e.g., Internet of Things devices, sensor nodes, mobile phones, high-end vehicles, etc. This system is comprised of two parts: a centralized control part with controllers, and a distributed data part. The controllers sit between the application layer and the physical device, performing and simplifying various network tasks. They act as bridges and provide a pathway for the upper layer to communicate with the device. Network devices must pass through the controller for messages to reach the application plane.

Our approach employs the utility-game theory and SDN to optimize task computation in a multilayer data flow processing system. The SDN controller consists of two main modules. The first module keeps track of the tasks and stores the mission data of all network nodes. It decides which tasks are processed by the edge for optimization. The second module is the edge server module, which contains data on the server's available memory and CPU, as well as its current load.

The key component of a smart manufacturing system is the SDN controller, which uses SDN protocols such as OpenDaylight and ONOS to centrally manage and allocate resources through sub-controllers. The production line consists of several network access points that act as switches and connect to the SDN controller via OpenFlow API to receive management information. In a smart manufacturing environment, devices are connected by wired or wireless networks to interact and send data processing requests to nearby access points, which in turn transmit the request situation and resource utilization to the SDN controller. The SDN controller then evaluates and predicts congestion and load status before sending control information to optimize routing and resource utilization. The private cloud computing center provides additional application services to supplement the edge service resources.

The suggested edge computing system based on SDN provides greater flexibility in defining network equipment forwarding functions through software programming, surpassing previous systems. The centralized control plane enables swift customization and software upgrades. Deploying edge computing nodes in a factory requires multiple access points to ensure timely detection, control, and execution of equipment. The QoS of the edge nodes may be affected by the transmission delay between the access points and the edge nodes.

The network layer includes different devices like switches, routers, gateways, and access points. These devices are important for building the edge computing network and providing advanced edge computing features. On the other hand, the edge computing layer consists of embedded devices that have specific ports, interfaces, and modules for data transfer and communications [24]. These devices cater to a diverse set of downstream devices. The edge computing node serves as an edge gateway, responsible for collecting, filtering, transferring, storing, analyzing, and processing device data while simultaneously uploading it to the cloud.

By deploying edge computing nodes in close proximity to factory equipment, several benefits can be achieved, including the utilization of a rule engine and function calculation, enabling efficient arrangement of application scenarios and business expansion. Additionally, it enables the execution of data processing tasks such as analysis, filtering, format conversion, and control logic at the edge, resulting in reduced network latency, minimized data loss, and improved real-time business continuity. Consequently, optimizing the deployment of edge computing nodes and carefully selecting the most suitable edge node for task computation are pivotal in enhancing the performance of real-time data processing.

III. Problem Statement

This section explains the topology of an edge network, which is represented by $G = (N, L)$, where N is a set of network nodes, L is a set of links, M is a set of edge nodes, and U is a utility function of edge nodes. The network consists of n nodes labeled as n_i, and each link between two nodes (i, j) has a bandwidth of B_{ij}.

Each edge node's computing capability is denoted by x_{ij}, where a value greater than zero ($x_{ij} > 0$) indicates that the edge node (j) serves the network node (i). A value of zero ($x_{ij} = 0$) indicates that it is not an edge computing node. In the edge network, there are s data source nodes (where s is less than n). Each node collects data at a specific rate of a_j per unit of time, and the response time for each edge node is t_j. The computing ability of edge nodes represented as x_j, is

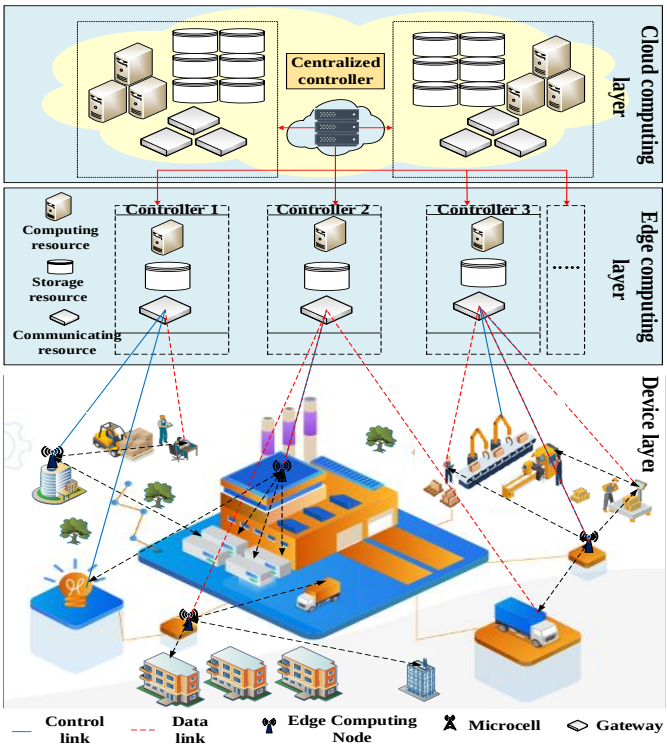

Figure 1. Network model.

always greater than zero, and the computing ability of network nodes cannot be negative.

In the proposed utility-game theory model, the primary aim is to maximize the overall reward achieved by all the edge nodes collectively. This game is mathematically expressed by Equation 2. To reach fact convergence, it is recommended that multiple players update their actions simultaneously during each iteration. However, to avoid interference among players, it is necessary to ensure that a player and its neighboring nodes do not update their actions simultaneously. Consequently, determining the neighboring nodes for each player becomes crucial.

To accomplish this goal, we create a graph that shows the connections between nearby nodes for each player. This graph is based on how close the network node is to its neighboring edge nodes. We use the Euclidean distance formula to calculate the normalized distance between the network node and its neighboring edge nodes.

We have a set called $\mathcal{A} = \{d_{11}, d_{22}, d_{33}, \cdots d_{ij}\}$ that contains the distances between the network node and its nearby edge nodes. Each distance, represented by d_{ij}, tells us how far the network node is from a specific neighboring edge node. Using this information, we can calculate the normalized distance as:

$$f_{\text{dis}}^{N} = \frac{d_{ij}}{max(A_i)} \tag{1}$$

where $max(A_i)$ compute the maximum distance from the set A_i. The problem can be stated mathematically as follows [25].

$$\max_{P1} \text{Payoff}_{tot} = \sum_{j=1}^{M} s_j \times \left(U_j - w_j \times {U_j}^2\right), \tag{2}$$

$$\arg\min_{\zeta} \sum_{m \in M} \zeta(m,n), \tag{3}$$

subject to

$$(\text{C1}) : \sum_{j=1}^{M} U_j = 1, \tag{4}$$

$$(\text{C2}) : \mathbb{P}(\sum_{m \in M} \zeta(n,m)) \leq P_m^{th} \tag{5}$$

$$(\text{C3}) : \sum_{k \in K} \zeta(n,m) \leq 1 \tag{6}$$

$$(\text{C4}) : \sum_{k \in K} \zeta(k,m) \leq U_j \quad \forall k \in K, \tag{7}$$

$$(\text{C5}) : \zeta(n,m) \in \{0,1\} \tag{8}$$

In the problem's formulation, the vector ζ consists of binary elements. When $\zeta(n,m) = 1$, it implies that the edge node m has been chosen to carry out a task computation for the network node n. Conversely, when $\zeta(n,m) = 0$, it means that edge node m is not selected. The utility function U_j signifies the current value of an edge node j prior to accepting a call request, while w_j represents the penalty weight for selecting an inappropriate edge node j. The penalty weight is crucial to avoid assigning high preference levels to unsuitable edge nodes, which could result in suboptimal solutions. One of the constraints that the optimization problem must satisfy is denoted as (C1), which guarantees that the payoff for each edge node is calculated by subtracting its cost from its reward.

In the utility-game model, the aim is to determine the set of policies that can achieve the highest possible payoff for each potential edge node. To ensure that an assigned edge node has enough computing capacity to handle a network node's task, constraint (C2) is introduced. This constraint considers the quality of service requirement and task size to determine whether an edge node can handle the task. Frequent task computation can increase overhead and delay in edge nodes. Constraints (C3), (C4), and (C5) restrict the assignment of each network node to a single edge node and limit the maximum number of edge nodes that each BS k ∈ K can serve simultaneously. Here, K denotes the set of BSs.

The preference level of a candidate edge node is usually higher when it has more available resources, while a higher penalty weight for an edge node results in a lower preference level. A candidate edge node that is closer to a network node is generally a better choice. To take this into account, the penalty weight w_j is determined based on the distance between the candidate edge node and the network node:

$$w_j = \begin{cases} 0 & \text{if } d_{ij} \leq d_{th} \\ \frac{(d_{ij} - d_{th})}{(c_j - c_{th})} & \text{if } c_{th} < d_{ij} \leq c_j \\ 1 & \text{if } c_j < d_{ij} \end{cases} \tag{9}$$

The optimization problem aims to maximize the total reward of all available edge nodes and is formulated in Eq. (9).

The distance between the network node i and edge node j is represented by d_{ij}, while c_j denotes the cell radius of edge node j. Pre-defined thresholds d_{th} and c_{th} are used to determine the penalty weight for an edge node based on its distance from the network node. If d_{ij} is less than or equal to d_{th}, the penalty weight is set to zero.

$$\begin{aligned} \max \quad & \text{Payoff}\,(U_1, U_2, ..., U_m) \\ \text{s.t.} \quad & \sum_{j=1}^{m} U_j = 1 \\ & U_1, U_2, U_3, ..., U_m \geq 0 \end{aligned} \tag{10}$$

The optimization problem formulated in Equation (10) will be tackled using the Karush-Kuhn-Tucker (KKT) conditions, as explained in [26]. The utility function value U_j for the jth edge node depends on coverage, power and distance between the edge node and the network node of edge nodes, as explained in the sequel. In the subsequent section, the edge node selection optimization problem will be transformed into a game theory framework.

IV. Edge Node Selection Model

This section presents a comprehensive explanation of the edge node selection model proposed in the study. The first subsection, Sec. IV-A introduces the utility game theory in manufacturing for controlling edge node services. Finally, a model is developed for the selection of edge nodes during the deployment of edge computing nodes.

A. Utility Game-Theory in smart manufacturing

The objective of the utility-game theory approach is to evaluate the worth of each potential edge node by computing its utility function. This can enhance the deployment of edge computing nodes and the collaborative game among edge nodes. The utility function enhances the quality of service and evaluates the utility value based on the requirements for enhancing the quality of service of the computation request task. The proposed edge node selection scheme employs a utility-game theory system to calculate the value of preference for each candidate edge node. The utility function considers the QoS improvement requirements of the task computation request. A multi-edge nodes viewpoint-based approach is used to calculate the preference value in a cooperative game for edge node selection. The proposed scheme aims to select the best edge node for a service request by integrating the utility values of all candidate nodes linearly and choosing the one with the highest value. The objective is to maximize the QoS benefit while minimizing cost and delay for a given state. To this end, a multiple edge node deployment model based on utility functions is designed. A game-theoretic auction model assigns a utility function to each edge node based on its features and task computing demands. The utility function is then used in the game model to choose the best edge node for the service process.

The selection of the best edge node is based on the computation of a utility function that is designed to accomplish the computation task. The optimal edge node should require a minimum cost of deploying edge computing nodes that have optimal distance and coverage, and have a power level that is above a certain threshold. Moreover, to ensure the QoS is maintained during the transition among edge nodes, a utility function is used.

The selection of edge nodes for task computing is considered a multi-objective optimization problem. To tackle this challenge, a utility game theory is devised to achieve optimal QoS for network nodes while minimizing the cost of deploying edge computing nodes. The game involves three key elements: game players, strategies, and payoffs. The set of game players is denoted as $\mathcal{M}$, comprising all potential edge nodes, each following its unique policy. The total number of strategies is denoted by g, represented by $\{U_1, U_2, .., U_j, .., U_m\}$, where U_j denotes the utility function value for the jth edge node. The objective is to minimize the overall utility function, formulated as:

$$\min \sum_{j=1,...,m} U_j \tag{11}$$

The term U_{j} can be decomposed into distinct components, i.e. [25],

$$U_{\mathrm{j}} = f_{\mathrm{cov}} \times f_{\mathrm{pow}} \times f_{\mathrm{dis}}^{N} \tag{12}$$

The connectivity and coverage of edge nodes are crucial for both requesting and executing network nodes to maintain a reliable connection during task execution and result transfer. If the mobile network node moves out of the coverage area of an edge node while processing a task, the task processing result must be sent to the cloud, resulting in an additional delay. The communication between network nodes and edge nodes is carried out through wireless links. The set of spectra that are orthogonal within the coverage area of an edge node is represented by k_j, ranging from 1 to K_j. By accessing these spectra, network nodes can access the data provided by the edge nodes. The binary spectrum decision of network nodes is denoted by $\{b.k_j = b.(1), ..., b.K_j\}$, where the value of $b.k_j$ is either 0 or 1. If a network node selects a particular edge node pair spectrum, k, then $b.k_j = 1$ and 0 otherwise. Each network node can select only one spectrum at a time, i.e., $b.k_j \leq 1$ for all k in K.

In order to calculate the coverage radius of an edge node, the coverage radius needs to be determined based on the minimum download network node coverage probability required. The formula for the coverage radius f_{cov} is given in Eq. (13), and it is defined as the maximum radius where the coverage probability for all mobile network nodes within that radius is above a given threshold. The calculation takes into account the distance of the edge node and its transmitting power.

$$r_j = \sqrt{\frac{P_{\mathrm{j}} G}{\lambda N_0 \varepsilon}}, \tag{13}$$

where P_{j} is the transmitter power (dBm), G is the antenna gain, λ is a parameter depending on the fading environment, N_0 is the power spectral density of the background noise, and ε is the minimum coverage probability. The coverage probability is determined as the probability that the received

signal power is greater than a threshold, which is related to the SINR requirement of the system.

To obtain the total power consumption of an edge node f_{pow}, we consider the idle and sleep modes. The sleep mode is enabled when there are no tasks to be executed or no scheduling activities are required. In the sleep mode, the power consumption of the edge node can be expressed as $Ps = P_{\mathrm{idle}}$, where P_{idle} is the power consumption of the idle state of the edge node. Therefore, the total power consumption of an edge node is given by the following equation:

$$P = \begin{cases} f_{\mathrm{pow}} & \text{if Active} \\ P_{\mathrm{idle}} & \text{if Sleep} \end{cases} \tag{14}$$

We need to compute the transmit power of the edge node in order to ensure that all network nodes within its coverage range can receive the signals. The transmit power is dependent on the distance between the edge node and the network node, the path loss, and the interference caused by other nodes. The transmit power can be calculated using the following equation:

$$P_{\mathrm{t}} = \frac{P_{\mathrm{r}} \cdot d_{ij}^{-\mu}}{\eta} \tag{15}$$

where P_{r} is the received power by the network node, d_{ij} is the distance between the edge node and the network node, μ is the path loss exponent, and η is the system efficiency.

The objective of optimizing the deployment of edge computing nodes is to reduce the delay in the edge network and minimize the cost associated with deploying computing resources. This goal is accomplished by minimizing the response time of data source nodes i.e.,

$$\min \sum_{j=1,\ldots,m} t_j \tag{16}$$

To achieve a balance between the two optimization objectives, weight parameters θ and λ are incorporated into the objective functions Eq. (16) and Eq. (11), respectively. This produces a new joint optimization objective, which can be represented as:

$$\min \left(\theta * \sum_{j=1,\ldots,m} t_j + \lambda * \sum_{j=1,\ldots,m} U_j\right) \tag{17}$$

To strike a balance between minimizing edge network delay and decreasing the cost of deploying edge computing resources, weight parameters θ and λ are employed. During the joint optimization process, these weight parameters serve as weights to control the relative importance of each objective, allowing for a more tailored optimization approach. By adjusting the values of θ and λ, it is possible to prioritize one objective over the other, depending on the specific needs and constraints of the system being optimized.

The given optimization problem is characterized as convex. It involves the minimization of a linear combination of two summations, each with coefficients θ and λ, respectively. The non-negativity of these coefficients preserves convexity, ensuring that any local minimum attained is also a global minimum. With no constraints, the problem exhibits the desirable property of convexity, indicating that it can be efficiently solved using convex optimization techniques and that the solution obtained is globally optimal.

The impact of these parameters on the optimization objective depends on the specific goal. For instance, if the objective is to minimize network time delay, then increasing θ and deploying more edge computing nodes is advantageous. Conversely, if the objective is to minimize the cost of computing resource deployment, then increasing λ and deploying fewer edge computing nodes are preferred. These approaches are appropriate for scenarios with different numbers of tasks, data transmission time constraints, and budget considerations. When both factors are considered to be equally important, a balance needs to be achieved between θ and λ. The aim of this study is to strike a balance between reducing-edge network delay and minimizing the cost of deploying computing resources, which is achieved by appropriately setting the values of θ and λ in the joint optimization objective.

The optimization process involves considering various factors such as the location of edge nodes, the computing capabilities of each node, and the data transmission requirements of the network. The goal is to identify the most suitable set of edge nodes that can provide the necessary computational resources while minimizing the cost of deployment. This involves evaluating the trade-offs between different factors, such as the distance between the edge nodes and the data sources, the processing power of the nodes, and the cost of deploying and maintaining the infrastructure. By taking into account all these factors, the optimization process can identify the most efficient and cost-effective deployment strategy for edge computing nodes. Ultimately, this will result in a low-latency network that can support a wide range of applications and services, while also being cost-effective to deploy and maintain.

The edge node selection problem can be modelled as a competitive game in a trading market, where the game players are the available edge nodes, and the policies represent the reaction of the game players to the network nodes' requirements, which are expressed in the weight of the parameters [27]. The payoff of each edge node is determined by its ability to compute the given task. To efficiently address this game, a scoring technique known as MEW (Multiple Evaluation Weighting) is utilized within a cooperative game model, as described in the work by Yoon et al. [28]. The MEW score function incorporates various criteria, including distance, power, and coverage, and the Analytic Hierarchy Process (AHP) method is employed to calculate the weights of network selection parameters based on task requirements. AHP is a valuable tool for calculating the weights of network selection parameters because it offers a structured, flexible, and systematic way to make decisions. It's especially useful when you have multiple criteria and subjective preferences to consider. AHP helps prioritize parameters, reach consensus, and make transparent and consistent decisions in network selection. Once an edge node is selected by the SDN controller, an agreement is established with the chosen edge node. This process ensures that the selected edge node guarantees optimal QoS to the network node. Overall, the edge node selection problem is critical for the efficient and effective deployment of edge

Algorithm 1 Edge Node Selection using AHP and Utility Theory

Require: List of edge nodes (edgeNodes)
Ensure: Best edge node for selection (bestEdgeNode)

```
function selectEdgeNode(edgeNodes):
    matrix ← generateGeneralizedMatrix(edgeNodes)
    weights ← calculateWeightsUsingAHP(matrix)
    payoffs ← calculatePayoffs(edgeNodes, weights)
    bestEdgeNode ← findEdgeNodeWithMaxPayoff(payoffs)
    return bestEdgeNode

function generateGeneralizedMatrix(edgeNodes):
    matrix ← createEmptyMatrix(len(edgeNodes), numFeatures)
    for i = 1 to len(edgeNodes) do
        for j = 1 to numFeatures do
            matrix[i][j] ← calculateFeatureValue(edgeNodes[i], j)
    return matrix

function calculateFeatureValue(edgeNode, j)
    {Extract the value associated with each feature (distance, power, coverage) pertaining to the edge node}
    featureValue ← edgeNode[j]
    return featureValue

function calculateWeightsUsingAHP(matrix):
    {Perform calculations using the AHP to determine the weights}
    return weights

function calculatePayoffs(edgeNodes,weights):
    payoffs ← empty list
    for i = 1 to len(edgeNodes) do
        payoff ← calculatePayoff(edgeNodes[i],weights[i]).....  Equation (10)
        payoffs.append(payoff)
    return payoffs

function calculatePayoff(edgeNode, weights):
    {Perform calculations to determine the payoff for the edge node}
    return payoff

function findEdgeNodeWithMaxPayoff(payoffs):
    maxPayoff ← -1
    bestEdgeNode ← null
    for i = 1 to len(payoffs) do
        if payoffs[i] > maxPayoff then
            maxPayoff ← payoffs[i]
            bestEdgeNode ← i
    return bestEdgeNode =0
```

computing infrastructure and requires careful consideration of various factors.

The Edge Node Selection using Game Theory is displayed in Algorithm 1. The proposed algorithm aims to select the best edge node for providing a service before the task execution begins. The algorithm takes a list of edge nodes as input and determines the best edge node for selection. The **generateGeneralizedMatrix** function creates a matrix that contains feature values for each edge node. Each row represents an edge node, and each column represents a specific feature (e.g., distance, power, coverage). The **calculateFeatureValue** function extracts the value associated with each feature for a given edge node. The **calculateWeightsUsingAHP** function is a placeholder for a step where AHP is used to calculate the weights of the features. AHP is a technique for determining the relative importance of criteria or features in decision-making. The **calculatePayoffs** function computes a payoff for each edge node based on the calculated weights from AHP. The payoffs represent how well each edge node performs considering the criteria and their weights. The **findEdgeNodeWithMaxPayoff** function identifies the edge node with the highest payoff, indicating the best choice based on the selected criteria. The process starts with edge nodes sending their initial offers to the controller, which then evaluates their ability based on the offers. The controller then negotiates with the edge nodes by distributing tasks, and each node responds based on their strategies represented by the weights of key parameters that impact data processing. Each edge node evaluates and sends its final offers based on the assigned tasks. Using a weighted matrix, the controller evaluates the final offers and selects the most favorable one. In the concluding phase, the edge node with the highest utility function score is chosen as the winner and entrusted with providing the service. Figure 2 shows the steps of the edge node selection model based on game theory to choose the best edge node for task processing.

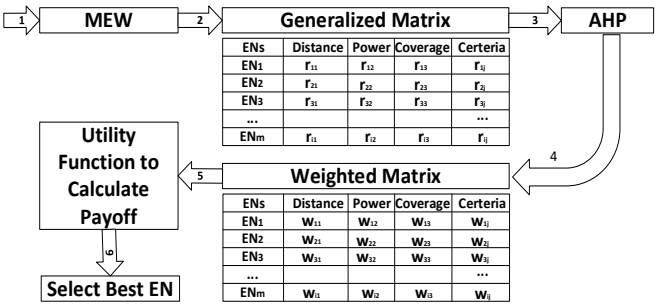

Figure 2. Edge node selection model.

The task of selecting the most suitable edge node for task computation can be represented as a matrix, where each row corresponds to a candidate edge node and each column corresponds to a specific metric, such as distance or power. The edge node's score, represented by S_j, is determined by multiplying the weight of each relevant metric, denoted as w_j, with its corresponding value. These weights represent the importance of each metric and are combined in a convex manner to compute the final score for each edge node, considering both advantageous and expensive metrics. The formula for the calculation of the score is given by:

$$S_j \triangleq \prod_{j=1}^{M} {x_{ij}}^{w_j} \tag{18}$$

Weight parameters $\mathrm{w_j}$ are allocated to each metric in the score evaluation, where x_{ij} represents the computing capability of the jth edge node serving the ith network node. The weight set forms a convex combination, with positive weights assigned to benefit metrics and negative weights assigned to cost metrics.

To evaluate the scores of the edge nodes, we compare the score obtained through MEW with the score of an ideal edge node. The edge node with the highest score is considered to be the best. The scores are calculated using a score function for each edge node. The value ratio r_j between the j-th edge node and the ideal edge node is defined in Equation (19). For cost metrics, a lower value is considered better.

$$r_j = \frac{j}{j^*} = \frac{\prod_{j=1}^{M} x_{ij}^{j}}{\prod_{j=1}^{M} \left(x_{ij}^*\right)^{w_j}} \tag{19}$$

where $0 \leq r_j \leq 1$.

The total score of the primary offer for each edge node is obtained according to:

$$U_j^{primary} = \sum_{j=1}^{m} r_j \tag{20}$$

The data fusion unit primarily performs the tasks of analyzing, calculating, and merging raw data to generate accurate attribute weights. The AHP technique can be employed to calculate the weighted matrix, which determines the relative weight w_j of the jth attribute in Equation (21). The weighted matrix incorporates the weights of various metrics that influence the edge node selection process, including distance, power, coverage, delay, and deployment cost. By summing up these weights, the relative weight of each attribute can be determined.

$$\sum_{j=1}^{m} w_j = w_d + w_p + w_c + w_{rs} + w_{cd} \tag{21}$$

The weight or importance of the metrics affecting the edge node selection process are denoted by w_d, w_p, w_c, w_{rs}, and w_{cd}, which respectively correspond to distance, power, coverage, delay, and deployment cost.

The game utility value U_j for the jth edge node is determined by multiplying the relative weight of task requirements w_j with the non-dimensional comparable data r_j. This synthesis is represented by:

$$U_j = \sum_{j=1}^{m} w_j r_j \tag{22}$$

where the total utility for each candidate edge node is calculated using the utility function U_j, and the optimal edge node is selected based on the edge node with the highest total utility score. This means that the index j^* of the optimal edge node is equal to the index j^* that maximizes the utility function U_j, which can be mathematically represented as follows:

$$j^\star \triangleq \arg\max_j U_j \tag{23}$$

V. Numerical Evaluation

This section is dedicated to evaluating the effectiveness of the proposed model in a simulated smart factory environment and comparing it with other benchmark algorithms. Here, we offer comprehensive information regarding the simulation environment settings and present the results obtained from our experimental evaluation. Our evaluation includes a comparative analysis of the performance between our model and other similar methods, namely Shortest Estimated Latency First (SELF) [29], Shortest Estimated Buffer First (SEBF) [30], and Random Deployment (RD). We assess their effectiveness in terms of deploying edge nodes cost, network delay, packet loss, and energy consumption. Our results demonstrate that the proposed algorithm performs exceptionally well in minimizing the total cost of deploying edge nodes, network delay, packet loss, and energy consumption.

The intelligent manufacturing process includes various functions and applications, resulting in multiple scenarios. For this study, we consider an existing smart factory scenario in a square environment. Based on the production process, smart devices are placed in corresponding positions on the grid, and WiFi sites are optimally placed based on the smart device locations. Using the models and algorithms presented in this article, we determine the optimal deployment location for edge servers and deploy them on the corresponding WiFi sites, as illustrated in Figure 1.

We consider an area of 1000m $\times$ 1000m which has 200 sensor nodes distributed randomly and 32 WiFi stations distributed regularly along the coordinate axis. Each sensor node is randomly assigned one of the 15 available services with the data size and CPU cycles for each service randomly chosen from a range. We deploy 15 edge server nodes, with each server's cost fixed between 1000 and 1500 due to variations in their computing resources. We set the threshold value to 800 for basic edge servers and 600 for fault-tolerant servers. The CPU resources of servers are within a range of 40 to 90, and the server throughput is set between 3 to 8. After a specific period, each edge node reconfigures to serve the maximum sensor node request according to the Poisson process.

We evaluated the performance of our proposed node deployment method by comparing it to three other methods: SELF, SEBF, and RD.

SELF operates by having a sensor node select the edge node with the shortest estimated latency based on calculations of latency between the sensor node and all reachable edge nodes, as shown in Figure 3. This approach aims to save energy. However, it faces challenges when an edge node is within the communication range of multiple sensor nodes. Simultaneous transmission can lead to data loss, as depicted in Figure 4, without considering data priority, which is problematic in critical situations. Nevertheless, in normal situations without critical events, the SELF method can extend the lifetimes of both sensor nodes and the network. SEBF, on the other hand, selects an edge node based on the smallest buffer occupancy, significantly reducing packet loss, as shown in Figure 4. However, it has a drawback when a sensor node is within the transmission range of multiple edge nodes. SEBF

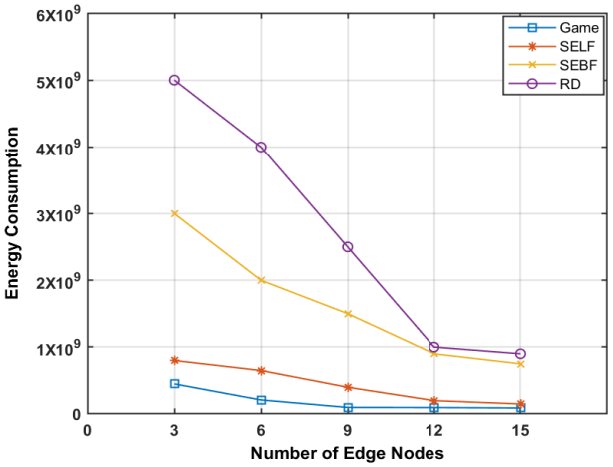

Figure 3. Number of edge nodes versus energy consumption (Number of sensor nodes = 200)

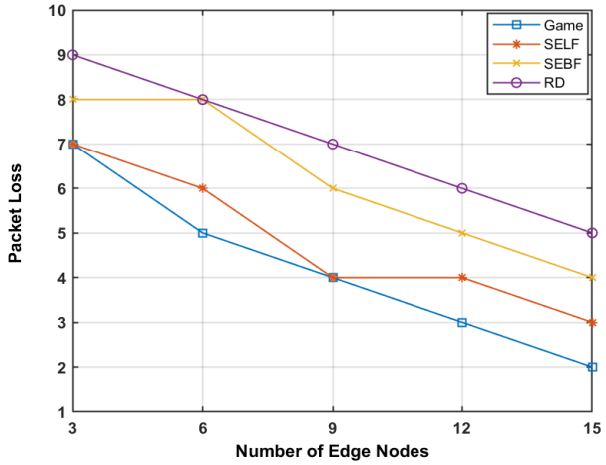

Figure 4. Packet loss versus number of edge nodes (Number of sensor nodes = 200)

may lead the sensor node to choose a more distant edge node with higher latency, as illustrated in Figure 3, resulting in increased energy consumption and reduced lifespans for both the sensor node and the network. While SEBF can be advantageous in critical situations, it may cause data loss in rare, life-threatening circumstances. Lastly, the RD method is straightforward to implement but has its drawbacks. There is a possibility that a sensor node selects an edge node with a higher buffer occupancy or higher latency compared to other equivalent edge nodes. The findings presented in Figures 3 and 4 demonstrate that our model effectively reduces energy consumption and mitigates packet loss compared to the SELF, SEBF, and RD approaches, providing a more reliable and efficient solution.

We also evaluated the proposed method for deploying edge computing nodes in smart manufacturing by considering two key factors: network delay and the cost associated with deploying computing resources. The comparison primarily focused on two aspects: the duration of data transfer from a device to an edge node and the time required to receive processing results back from the edge node. Additionally, the cost of deploying computing resources for the overall task computation process was also considered in the evaluation. The objective was to achieve a cost-effective deployment of computing resources while ensuring that all edge computing functions were met. This was achieved by finding a balance between network delay and the cost of deploying computing resources and optimizing the deployment of edge computing nodes throughout the manufacturing space. For example, if a factory's edge computing functions could be met with three nodes, any additional nodes would be unnecessary and would increase costs.

In our simulation, we deliberately chose specific values for two key parameters, θ and λ, setting them to 1000 and 0.2, respectively. This deliberate choice aimed to strike a delicate balance between the impact of network delay and the deployment cost of computing resources within the edge computing node deployment cost function. These parameter selections were instrumental in shaping the behavior of our deployment model.

The impact of these parameter choices became evident when examining the variation in network delay, as illustrated in Figure 5, across different algorithms concerning the number of deployed edge computing nodes. As more nodes were deployed, each node had a reduced workload in terms of production equipment assignments, leading to a notable reduction in network delay across all considered methods. Intriguingly, our game theory-based method demonstrated superior performance in terms of minimizing network delay compared to the SELF and RD methods when deploying between 3 to 9 edge nodes. However, as the number of edge nodes exceeded 9, the network latency difference between our method and SELF gradually diminished. With a higher number of edge nodes, the request-response time value appeared to stabilize at approximately 300–400 ms.

This observation underscores the effectiveness of our game theory-based approach in optimizing edge computing node deployment for reduced network delay, particularly within a certain range of edge node counts. However, it also highlights the diminishing returns associated with further increases in edge node deployment, where network latency tends to stabilize. This nuanced understanding is valuable in tailoring edge computing strategies to specific manufacturing scenarios where network delay and deployment cost considerations are paramount.

The deployment cost of various methods and the number of edge nodes is shown in Figure 6. The proposed game theory method had the best performance in terms of deployment cost. In SELF, SEBF and RD, for the number of edge nodes greater than 6 nodes, the network has more available resources and the edge nodes can meet most of the requests locally. As more edge nodes were added, the proposed game theory method performed better than other methods in terms of network delay and deployment cost. The method selected an edge node to meet a time gain threshold of less than 8%, resulting in the lowest deployment cost and network delay of 0.6 seconds compared to the other methods. Overall, the proposed method

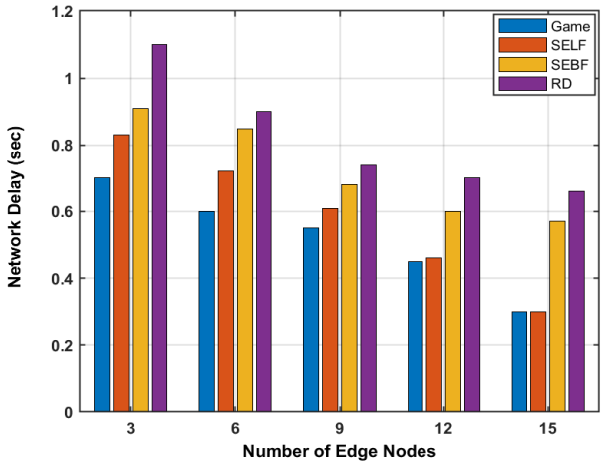

Figure 5. Number of edge nodes versus network delay.

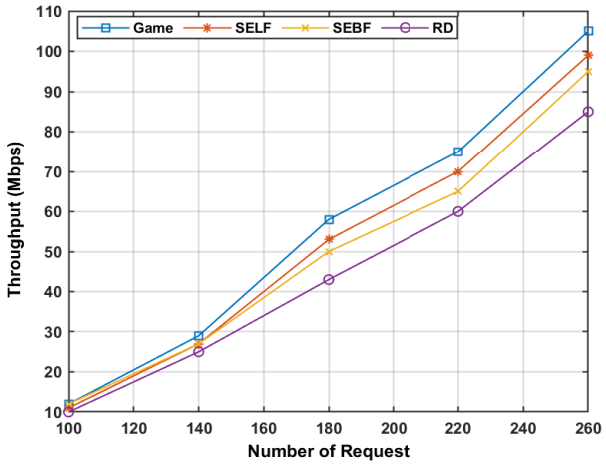

Figure 7. Throughput versus number of requests.

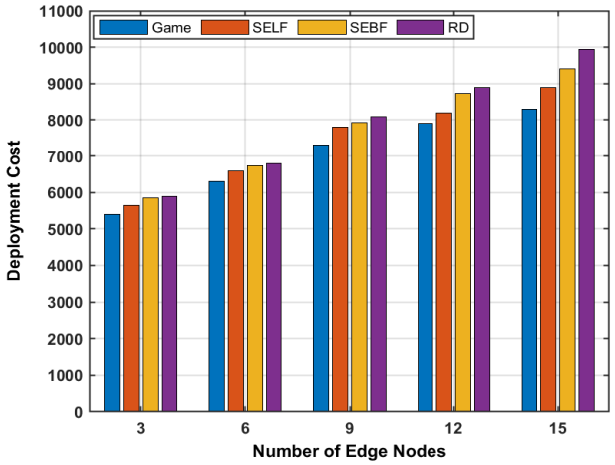

Figure 6. Number of edge nodes versus deployment cost.

outperformed all other approaches in terms of both network delay and deployment cost.

In Figure 7, we provide a visual representation of the relationship between the number of incoming requests and the resulting throughput performance. What immediately captures attention is the consistent and remarkable superiority of the network design proposed in this study when compared to three benchmark schemes—SELF, SEBF, and RD.

To be more specific, our proposed network demonstrates a respective 6% improvement compared to the SELF approach, a 10.5% enhancement compared to the SEBF approach, and a substantial 23.5% boost in throughput performance compared to the RD approach when the number of requests reaches 260. This significant performance advantage indicates not only the increased capacity but also the efficiency of our proposed network configuration in handling higher request loads. It suggests that our network architecture is well-equipped to process a greater number of requests in a more efficient and expedient manner, making it a robust choice for scenarios where high-demand processing is required. This enhanced throughput can translate into improved user experiences, reduced latency, and increased overall system efficiency, making our network design an attractive and practical solution for various real-world applications.

VI. Conclusions and Future Directions

We presented a technique for deploying edge computing nodes in a system architecture, aiming to strike a balance between deployment cost, network delay, and other factors such as the spatial distribution of devices, device functions, and edge node computing capacity. To accomplish this, a game theory algorithm was proposed to find the best edge computing node for deployment. We compared this method with three other techniques: SELF, SEBF, and random deployment. The results of the experiments demonstrated the superiority of the proposed method over the other three benchmark schemes. in terms of network delay and computing resource cost. This approach represents a valuable and promising solution for implementing Industry 5.0 and improving the real-time data processing of a device. In future work, our aim is to delve deeper into the development of the learning decision-making mechanism rooted in the utilization of this deployment approach, with the overarching goal of significantly augmenting device performance. This research will involve an in-depth examination of the learning decision-making process, leveraging the insights gained from the deployment method to make more informed and optimized choices. By integrating this enhanced decision-making mechanism into our framework, we anticipate achieving notable improvements in device performance across various applications and scenarios.

Acknowledgment

This work was supported in part by the U.K. Engineering and Physical Sciences Research Council under Grant EP/X013162/1.

References

[1] P. O. Skobelev and S. Y. Borovik, "On the way from industry 4.0 to industry 5.0: From digital manufacturing to digital society," *Industry 4.0*, vol. 2, no. 6, pp. 307–311, 2017.

[2] V. Özdemir and N. Hekim, "Birth of industry 5.0: Making sense of big data with artificial intelligence,"the internet of things" and next-generation technology policy," *Omics: a journal of integrative biology*, vol. 22, no. 1, pp. 65–76, 2018.

[3] N. Jafari, M. Azarian, and H. Yu, "Moving from industry 4.0 to industry 5.0: what are the implications for smart logistics?" *Logistics*, vol. 6, no. 2, p. 26, 2022.

[4] 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.

[5] L. Lv, Z. Wu, L. Zhang, B. B. Gupta, and Z. Tian, "An edge-ai based forecasting approach for improving smart microgrid efficiency," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 7946–7954, 2022.

[6] L. Yang, Y. Lu, J. Cao, J. Huang, and M. Zhang, "E-tree learning: A novel decentralized model learning framework for edge ai," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 290–11 304, 2021.

[7] X. Feng, J. Wu, A. K. Bashir, J. Li, A. Shen, and M. D. Alshehri, "Vulnerability-aware task scheduling for edge intelligence empowered trajectory analysis in intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4661–4670, 2023.

[8] S. A. Soleymani, S. Goudarzi, M. H. Anisi, Z. Movahedi, A. Jindal, and N. Kama, "PACMAN: Privacy-preserving authentication scheme for managing cybertwin-based 6G networking," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4902–4911, 2021.

[9] S. Goudarzi, M. H. Anisi, A. H. Abdullah, J. Lloret, S. A. Soleymani, and W. H. Hassan, "A hybrid intelligent model for network selection in the industrial internet of things," *Applied Soft Computing*, vol. 74, pp. 529–546, 2019.

[10] J. Wan, S. Tang, Q. Hua, D. Li, C. Liu, and J. Lloret, "Context-aware cloud robotics for material handling in cognitive industrial internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2272–2281, 2017.

[11] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6g networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5098–5107, 2020.

[12] C. F. Durach, J. Kembro, and A. Wieland, "A new paradigm for systematic literature reviews in supply chain management," *Journal of Supply Chain Management*, vol. 53, no. 4, pp. 67–85, 2017.

[13] N. Iqbal, A.-N. Khan, A. Rizwan, F. Qayyum, S. Malik, R. Ahmad, D.-H. Kim *et al.*, "Enhanced time-constraint aware tasks scheduling mechanism based on predictive optimization for efficient load balancing in smart manufacturing," *Journal of Manufacturing Systems*, vol. 64, pp. 19–39, 2022.

[14] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, "Adaptive transmission optimization in sdn-based industrial internet of things with edge computing," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1351–1360, 2018.

[15] J. Wan, B. Chen, M. Imran, F. Tao, D. Li, C. Liu, and S. Ahmad, "Toward dynamic resources management for iot-based manufacturing," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 52–59, 2018.

[16] J. Wan, S. Tang, D. Li, M. Imran, C. Zhang, C. Liu, and Z. Pang, "Reconfigurable smart factory for drug packing in healthcare industry 4.0," *IEEE transactions on industrial informatics*, vol. 15, no. 1, pp. 507–516, 2018.

[17] Z. Zhou, M. Shojafar, M. Alazab, and F. Li, "Iecl: an intelligent energy consumption model for cloud manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8967–8976, 2022.

[18] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4702–4711, 2018.

[19] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. Rodrigues, and M. Guizani, "Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay," *IEEE communications magazine*, vol. 56, no. 2, pp. 44–51, 2018.

[20] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[21] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2015.

[22] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4712–4721, 2018.

[23] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, and A. Celesti, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4225–4234, 2019.

[24] A. Majeed, Y. Zhang, S. Ren, J. Lv, T. Peng, S. Waqar, and E. Yin, "A big data-driven framework for sustainable and smart additive manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 67, p. 102026, 2021.

[25] S. Goudarzi, M. H. Anisi, D. Ciuonzo, S. A. Soleymani, and A. Pescapé, "Employing unmanned aerial vehicles for improving handoff using co-operative game theory," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 2, pp. 776–794, 2020.

[26] C. A. C. Cortez and J. C. Pinto, "Improvement of Karush–Kuhn–Tucker conditions under uncertainties using robust decision making indexes," *Applied Mathematical Modelling*, vol. 43, pp. 630–646, 2017.

[27] T. L. Saaty, "Decision-making with the AHP: Why is the principal eigenvector necessary," *Elsevier European Journal of Operational Research*, vol. 145, no. 1, pp. 85–91, 2003.

[28] K. P. Yoon and C.-L. Hwang, *Multiple attribute decision making: an introduction*. Sage publications, 1995, vol. 104.

[29] T. Rahman, X. Yao, G. Tao, H. Ning, and Z. Zhou, "Efficient edge nodes reconfiguration and selection for the internet of things," *IEEE Sensors Journal*, vol. 19, no. 12, pp. 4672–4679, 2019.

[30] A. Alagha, S. Singh, R. Mizouni, A. Ouali, and H. Otrok, "Data-driven dynamic active node selection for event localization in iot applications-a case study of radiation localization," *IEEE Access*, vol. 7, pp. 16 168–16 183, 2019.