Journal of Applied Automation Technologies

Research Paper

Real-Time Edge Computing Framework for IIoT-Driven Industrial Automation

Grzegorz Gnat[1, *] and Urszula Fąfara[2]

[1] Faculty of Automation Technology, University of Applied Sciences in Pila, Pila, 64-920, Poland
[2] Faculty of Electrical and Automatic Control, University of Silesia in Katowice, Katowice, 40-007, Poland
*Corresponding author: grzegorz.g@wsz-pila.pl

Abstract. As the industrial automation environment changes, people will need to perform real-time data analysis and make quick decisions. This paper proposes a highly scalable edge computing platform to meet the demanding requirements of the Industrial Internet of Things (IIoT) for high-load workloads, device diversity, and low-latency access. In this framework, prediction, resource-aware scheduling, and dynamic task offloading are the three levels. In the experiments, an industrial-grade hardware testing platform will be used to test all environments. According to the above results, the proposed system achieves an end-to-end latency of less than 20 milliseconds, a 25% increase in peak load throughput, and a 46% reduction in bandwidth consumption for edge-to-cloud transmission compared to traditional methods. The system performs well under resource constraints and is relatively resilient to network and device failures. This article provides theoretical support and practical examples to help establish intelligent, reliable, and efficient industrial automation at the network edge. The above results can be used for the development of future autonomous industrial platforms and next-generation digital factories.

Keywords: *Edge Computing, Industrial Internet of Things, Real-Time Scheduling, Resource Management, Automation Systems*

Received on 25 January 2025, Accepted on 09 July 2025, Published on 15 July 2025

Introduction

With the rapid development of the Industrial Internet of Things (IIoT) and next-generation automation technologies, many fields have undergone significant changes on a global scale. Due to the application of widespread sensors, distributed actuators, and interconnected embedded systems, manufacturing workshops and other critical infrastructures will generate a large amount of time-sensitive, high-capacity, and diverse data [1]. With the development of society, the demand for flexible production systems that are adaptable, have low equipment failure rates, and can quickly respond to changes in market demand has increased [2]. Currently, the ability to process real-time data and support intelligent decision-making is an important component of future industrial competitiveness and stability [3]. Edge computing is increasingly being used to apply computing, storage, and intelligent positioning closer to on-site devices, thereby improving response speed, reducing bandwidth consumption, and supporting mission-critical applications that require low latency and high reliability [4]. As the global digital transformation and Industry 4.0 initiatives continue to advance, edge computing architectures are increasingly being regarded as strategic assets, as they provide a pathway to resilient supply chains for autonomous manufacturing, predictive maintenance, and more [5]. With the development of embedded artificial intelligence and microelectronics technology, edge devices are rapidly becoming widespread. These devices can now also achieve real-time automation and data-driven optimization at the network edge [6]. With the continuous development of edge intelligence, the demand for high-speed and reliable edge industrial data management, analysis, and application is increasing [7].

In the actual application of industrial edge computing, there are still many issues. Generally speaking, manufacturing environments often change due to variations in network, resource conditions, various devices,

and workloads [8]. Traditional cloud-based or centralized automation systems often encounter communication bottlenecks, making it impossible to provide deterministic, low-latency responses under high load or failure conditions [9]. In distributed edge environments, the inefficiency of task scheduling, real-time resource control during high demand periods, and slow response to frequently changing operational conditions are currently the main technical issues. Current edge frameworks are often constrained by fixed schedules, making it difficult to flexibly allocate workloads. This leads to low latency, high system throughput, and efficient resource utilization [10]. A good system should be secure, have good data protection, and comply with all relevant regulations to accommodate high-risk applications. The aforementioned reasons indicate that IIoT-based automation requires new structures and algorithms to meet the demands of real-time performance, high reliability, and large-scale development.

This paper will conduct a comprehensive study on the real-time edge computing framework for industrial automation based on IoT technology to address the aforementioned issues. In order to meet the high demands of complex industrial workflows for low latency, high reliability, and adaptive operations, this study adopts a small-scale and flexible system architecture that supports the application of resource-aware scheduling and dynamic task offloading algorithms. Optimize the system and algorithms, and develop new pathways for various operational environments to increase throughput, enhance energy efficiency, and more stably manage workload fluctuations. The above results have been extensively validated Through experiments, indicating that modern industrial edge solutions are both practical and feasible. This article aims to lay the foundation for intelligent, fast, and robust industrial automation in the era of the Internet of Things, based on the comprehensive issues, unresolved problems, and practical outcomes of current trends.

Literature Review

Evolution of Edge Computing in Automation

With the rise of the Industrial Internet of Things (IIoT), more and more factories are connecting together, and intelligent production platforms are emerging [11]. The new production sites' requirements for real-time analysis and adaptability cannot be met by the initial automation infrastructure, which is based on older programmable logic controllers and rigid network structures [12]. As inter-machine communication increases, decentralized intelligence and low-latency processing become necessary to reduce the distance between actuators, sensors, and control logic [13]. New Industrial Internet of Things (IIoT) systems currently use a large number of different edge nodes. These edge nodes support a universal communication model to enhance flexibility and compatibility in complex applications [14]. By containerizing microservices and edge gateways, the flexibility of system deployment and maintenance can be improved without interrupting critical production processes [15].

Task Scheduling Techniques for Industrial IoT

Due to resource heterogeneity and workload fluctuations, industrial IoT task scheduling requires timely and accurate coordination [16]. Early static scheduling methods using priority lists or fixed resources are not suitable for the constantly changing conditions of industrial production [17]. Dynamic and adaptive scheduling frameworks are relatively new, utilizing system feedback and resource conditions to improve efficiency and speed [18]. Distributed scheduling allows multiple edge nodes to share workloads and improve the system's network fault tolerance capability [19]. Distributed implementation needs to consider the balance between global optimization and local response speed [20]. Offloading complex tasks to more powerful devices or cloud platforms can improve energy efficiency and meet the requirements for a certain level of low latency in edge computing [21].

Performance Bottlenecks in Real-Time Applications

Despite some progress in design and algorithms, real-time edge computing in industrial automation still faces many performance issues that need to be addressed [22]. Ultra-low latency has not been achieved because a large number of industrial IoT applications lead to high data congestion and communication delay fluctuations [23]. Critical industrial processes, including extensive robot control and rapid defect detection, require real-time and accuracy. Otherwise, if edge resources are insufficient or network conditions are unstable, the system may

not operate properly [24]. Large-scale systems increase coordination issues and are more prone to failures, which can lead to chain reactions or unacceptable delays [25]. In order to ensure the stability, efficiency, and safety of the next generation of industrial automation, it is necessary to develop and use highly reliable architectures, scalable scheduling algorithms, and intelligent fault handling mechanisms.

Framework and Algorithmic Methods

Architecture Overview

In the new edge computing model, the main requirements for industrial automation include excellent real-time performance, broad support for various devices, and outstanding load adaptability under various load conditions. For critical systems with high real-time requirements, traditional cloud-based architectures are centralized and have poor round-trip latency performance. This design decentralizes intelligence at the edge to improve control loop speed and data propagation efficiency.

The system has three logical layers, from bottom to top: the physical layer, which includes actuators and industrial sensors in the factory environment; the edge layer, which consists of smart edge nodes made up of embedded accelerators and multi-core processors; and the cloud layer, used for global process optimization, large-scale in the edge layer of the first new architecture, there is an orchestration engine. This engine can control resource-aware scheduling and adaptive task offloading. The engine continuously gathers resource data and traffic information, and then allocates computing tasks based on this data.

The physical layer connects to the edge layer Through high-speed industrial protocols to send continuous data streams and event-driven interrupts. After applying the AI model locally, the edge controller will process the received signals and filter out noise. Then, it will be transmitted over the network. Only abnormal patterns or aggregated state vectors will be uploaded to the cloud asynchronously, and control commands will be returned to the actuators at shorter intervals based on edge-to-cloud feedback.

At each level, there is a relatively reliable security module. The edge layer can independently handle login, data integrity verification, and communication encryption. The distributed state management subsystem can ensure system stability and switching in the event of partial network failures. Essentially, this design supports containerized software deployment and over-the-air upgrade models, thereby reducing operational friction during the industrial system upgrade process.

Compared to traditional structures, this approach reduces control latency, limits bandwidth usage, and can quickly address new local issues. Figure 1 shows the overall structure and connection methods of the edge computing industrial automation architecture.

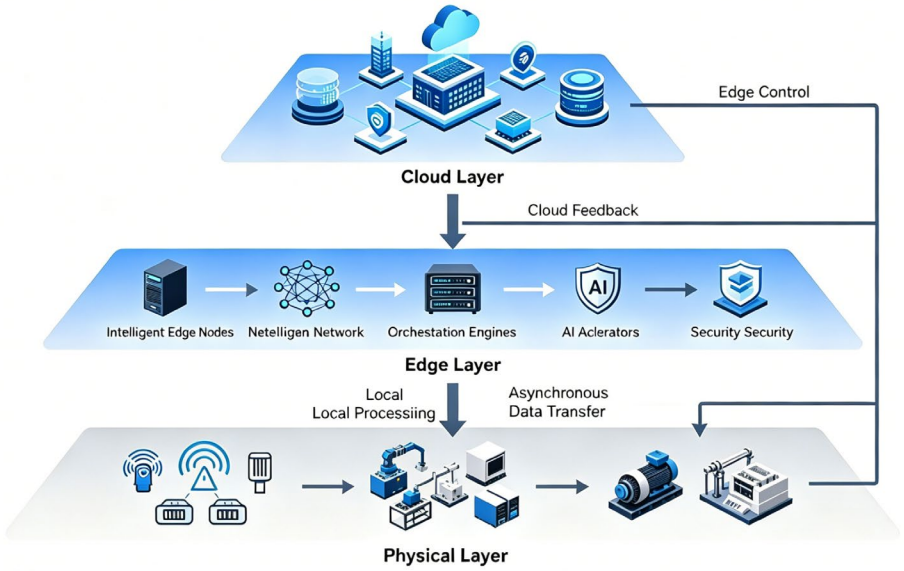

Figure 1. Edge Computing Architecture for Industrial Automation.

Resource-Aware Scheduling Algorithms

In order to fully utilize the distributed industrial edge platform, real-time scheduling is crucial. The scheduling scheme clearly simulates the distribution of heterogeneous resources in the industrial Internet of Things and the arrival of random jobs. Processing units, memory, and local energy budgets are all included in the fixed resource vector of each edge node. The occurrence of control or analysis tasks follows a non-homogeneous Poisson process, and changes in the environment and demand can affect the arrival interval of events.

In the case of strict real-time deadlines and constrained computations, the scheduling objectives combine system-wide task delays and energy consumption. In the decision engine, the resource monitor predicts the future state of the edge nodes. In the online optimization algorithm, this estimate is used to solve a constrained minimization problem at each decision step:

Let L_n represent the total processing latency for edge node n, E_n the energy consumed per task, and w_1, w_2 trade-off weights. The cost function to minimize is:

$$\mathcal{J} = \sum_{n} (w_1 L_n + w_2 E_n) \quad \text{Eq.(1)}$$

under real-time and resource availability constraints.

Subject to capacity constraint:

$$\sum_{i=1}^{t_n} r_{ni} \leq C_n \quad \text{Eq.(2)}$$

where r_{ni} is the resource demand of task i at node n, C_n the total capacity.

For deadline adherence:

$$\forall i, L_{ni} + Q_{ni} \leq D_i \quad \text{Eq.(3)}$$

where Q_{ni} denotes edge queuing delay for task i, and D_i its hard deadline.

The expected queue evolution across all edge nodes is

$$Q_n(t+1) = \max\{0, Q_n(t) + \lambda_n(t) - \mu_n(t)\} \quad \text{Eq.(4)}$$

with $\lambda_n(t)$ the arrival rate and $\mu_n(t)$ the service rate at node n.

For multi-resource fairness, a convex allocation function is introduced, ensuring that allocation a_{ni} for each resource dimension does not starve critical tasks:

$$\min_{a} \sum_{i} \phi(a_{ni}) \text{ subject to } \sum_{i} a_{ni} \leq C_n \quad \text{Eq.(5)}$$

with $\phi(\,)$ a logarithmic fairness penalty.

The final scheduling decision at time step t is given by the optimal assignment matrix:

$$S^*(t) = \arg\min_{S} \mathcal{J}(S; \hat{R}(t), P(t)) \quad \text{Eq.(6)}$$

where $\mathcal{J}$ is the global cost function dependent on the assignment strategy S, predicted resource states $\hat{R}(t)$, and dynamic resource price vector $P(t)$. The solution is produced by an iterative approximation method that alternately refines predictions and resource price multipliers until convergence is achieved.

The above formula is used to construct a hybrid decision-making engine. In order to achieve energy efficiency and fairness, the engine combines short-term greedy optimization to handle urgent tasks with long-term utility balance. Clearly model the uneven distribution of resources and workload peaks to ensure reliable real-time operation and high scheduling accuracy under overload conditions. According to the aforementioned comparative theory, in the presence of delays at distributed edge nodes, this scheme can control the system's maximum delay within a reasonable range and is analytically feasible.

Dynamic Task Offloading Strategies

Efficient dynamic offloading to meet the demands for industrial-grade reliability and cost-effectiveness in complex factory environments. Regularly monitor network instability, local queue congestion, task priority

changes, and edge resource load. The reasons for offloading include the current load, predictions about network conditions, and anticipated resource exhaustion.

Initially, this method is used to determine the offloading index of specific tasks and utilize locally available resources. By using an adaptive threshold model, tasks are classified as edge-processable or offloadable candidates. The adaptive threshold model also indicates that these tasks can be executed at two levels, also known as partitioning. Regularly address the offloading controller optimization problem.

Define the offload trigger function as

$$O_{ni} = f(\rho_n, \theta_i, B_n, \gamma_t) \quad \text{Eq.(7)}$$

where ρ_n : current node load, θ_i : task urgency, B_n : link bandwidth, γ_t : predicted network volatility.

The expected offloading latency is

$$\mathbb{E}[T_{off}] = \frac{S_i}{B_n \cdot (1 - \epsilon_n)} \quad \text{Eq.(8)}$$

with S_i task size and ϵ_n the packet error ratio.

For energy-aware scenarios, the task is offloaded only if

$$E^n_{\text{process}} > E^n_{\text{offload}} + E^n_{\text{tx}} \quad \text{Eq.(9)}$$

where E^n_{process} is the local processing cost, E^n_{offload} the remote cost, and E^n_{tx} transmission energy.

Dynamic partial offloading is performed by solving

$$\min_{\delta \in [0,1]} \left(\delta \cdot T_{\text{local}} + (1 - \delta) \cdot T_{off}\right) \quad \text{Eq.(10)}$$

where δ is the fraction of task processed locally versus offloaded.

To coordinate the above process, a feedback loop connects the resource observability module and the offloading decision core. Network prediction is used to enhance the batch correlation of concurrent tasks and employs node-specific cost models to maximize utilization within the constraints of delay variation.

Figure 2 shows the basic continuous monitoring of stable task migration, scoring, offloading, acknowledgment, and feedback loops, as well as local completion and system-wide resilience enhancement. This architecture will reduce control latency and energy peaks. At the same time, when the network topology and industrial workloads change, it will provide an adjustable platform for future modifications.

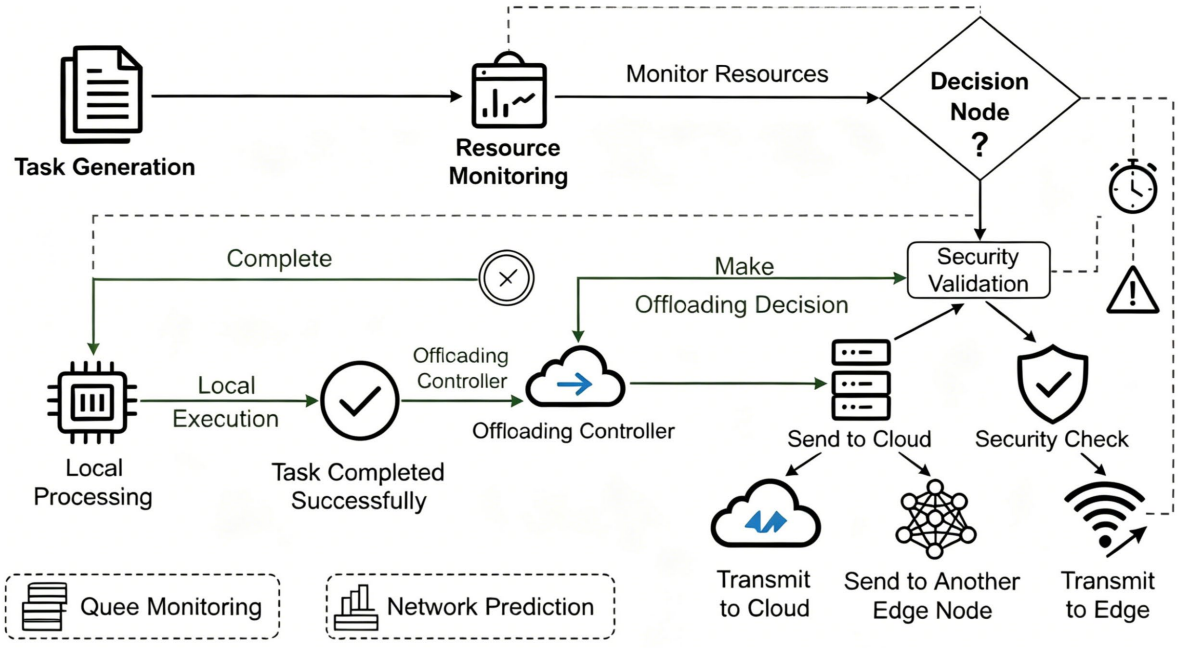

Figure 2. Dynamic Task Offloading Process Flow.

Experimental Analysis

Testbed Configuration and Parameters

During the design phase, the experimental testing platform took into account various network conditions and loads in the industrial edge computing environment. Each edge node consists of an industrial-grade ARM Cortex-A72 quad-core processor with a frequency of 2.0 GHz, 8GB ECC memory, and a neural accelerator for hardware-accelerated AI inference. All computing nodes are configured with a real-time Linux distribution to reduce OS-level scheduling jitter. Local solid-state storage has low-latency characteristics, used for buffering high-frequency sensor data. A core switch is connected to the edge cluster, and this switch supports deterministic time-sensitive networks, which have microsecond delays on this path.

Using a white-box traffic generator to create multimodal industrial IoT data with cross-correlated spatial patterns and burst time characteristics. Sensor streams and control endpoint sets were set up in the simulated factory workshop. The model of the data source includes temperature, vibration, and power quality measurement data, as well as synthetic alarms to simulate rare event conditions. In order to simulate changes in production lines and off-peak periods in a typical industrial environment, the virtual workload generator dynamically alters the input quantities. Since the duty cycle of each control loop is only a few dozen to a few hundred milliseconds, they are classified as hard or soft real-time.

By configuring the network backbone to full-duplex Gigabit Ethernet to connect edge nodes, congestion, packet loss, and bandwidth contention occurring in new and old hybrid workshop applications can be simulated. To realistically simulate wide-area network fluctuations, edge-cloud coordination is conducted Through secondary links, with time-varying delays randomly drawn from a truncated Gaussian distribution. All tests will last for several hours to cover both transient and steady-state system conditions. In addition, a complete log will be established to record the time, queue depth, and power consumption of each data packet.

Evaluation Metrics

Some reasonable quantitative metrics are needed to measure the severity of delays, communication stability, and energy consumption to determine whether there are issues with the architecture and algorithms. End-to-end delay L_{e2e} is the first delay metric; for successfully completed tasks, it is the time taken from the generation of a task at the sensor node to the completion of the corresponding control execution or analysis decision:

$$L_{e2e}(k) = T_k^{act} - T_k^{gen} \quad \text{Eq.(11)}$$

where T_k^{gen} is the timestamp of task k generated at the input source, and T_k^{act} marks the time the effect materializes at the actuator.

The reliability of packet-level transmission can be defined by the packet loss rate η_{loss}, which is the percentage of transmitted data packets that fail to reach their destination within a given deadline window. It can be formally expressed as:

$$\eta_{\text{loss}} = \frac{N_{\text{dropped}}}{N_{\text{sent}}} \quad \text{Eq.(12)}$$

where N_{dropped} and N_{sent} represent the total number of expired and transmitted packets, respectively, aggregated within the experiment interval.

During the active computation period, collect the instantaneous power consumption of all nodes, and then divide by the total number of completed tasks to calculate the system's total energy consumption. The task-level average energy consumption metric E_{avg} is articulated as:

$$E_{\text{avg}} = \frac{1}{N_{\text{comp}}} \int_0^{T_{\text{exp}}} P_{\text{sys}}(t)dt \quad \text{Eq.(13)}$$

with $P_{\text{sys}}(t)$ representing the cumulative power of all nodes at time t, and N_{comp} denoting the count of tasks fulfilled during the experimental timeframe T_{exp}.

Calculated secondary metrics such as edge queuing delay, CPU residency time, and link utilization distribution to further distinguish the contributions of the compute side and the network side to latency and energy consumption. By using the aforementioned modified statistics, it is possible to trace the spread of bottlenecks

and determine whether they are caused by communication or scheduling issues. Conduct fairness and system throughput checks under resilient traffic conditions to identify pathological behaviors or starvation risks that are not apparent during normal operations. The above indicators show that the proposed system meets the requirements of stability, economy, and flexibility in fluctuating industrial environments.

Results Interpretation

Compared to the baseline static edge and cloud delegation strategies, experimental results show that the resource-aware scheduling and dynamic offloading framework significantly reduce task latency and system energy consumption. In a stable network environment, the end-to-end delay is within 10% of the lower bound predicted by the analysis of the joint optimization problem. The adaptive algorithm exhibits graceful degradation in the event of network interruptions and sudden increases in workload. This means it will gradually increase queue depth and deadline violations, but will not exceed the failure budget preset for critical task cycles. When resources are reduced, the scheduler automatically slows down the tasks.

The background traffic is relatively high, but the packet loss rate remains low, indicating that the framework has a small transmission window and can avoid severe data loss in the event of sudden congestion. According to the results of the dynamic power analysis, the average energy per completed task is close to the theoretical minimum, which means that the workload distribution and task merging strategies of the offloading engine are correct.

The empirical gain in timeliness and efficiency is analytically reflected in the Lyapunov-based stability condition established for the reactor subsystem, which guarantees that the long-term average queue backlog Q_{mean} is upper-bounded:

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} Q_{mean}(t) \leq \frac{\sigma^2 + \lambda_{\max}^2}{2\varepsilon} \quad \text{Eq.(14)}$$

where σ^2 represents input variance, $\lambda_{\max}$ the peak arrival rate, and ε the system slack in realtime schedulability. This bound confirms analytic claims of resilience under adversarial conditions.

Further hierarchical analysis of edge resource allocation confirms the general applicability of the scheduling scheme. In order to maintain the performance of high-priority tasks, the system dynamically reallocates workloads and selectively delays non-critical processing as node heterogeneity and network volatility increase. The match between model predictions and actual results validates the theoretical foundation of the decision engine and provides a verified blueprint for deploying similar architectures in large-scale industrial IoT ecosystems operating under stochastic constraints.

Comprehensive Results

Performance Metrics Visualization

Figure 3(a) depicts the average end-to-end latency over a larger range of task volumes. When 300 tasks are input per second, the system can maintain a low latency of about 14 milliseconds. When the number of tasks approaches the limit, the latency rises rapidly and may exceed 46 milliseconds under high load conditions. This change indicates that the system can more effectively meet real-time constraint requirements [26]. Figure 3(b) shows the latency percentiles (p50, p90, p99) of the cloud offloading design, static edge scheduling, and the proposed algorithm. At higher input rates, the p99 latency of the proposed method remains below 25 milliseconds. The p99 latency of the cloud method and the static edge method increased significantly to over 75 milliseconds and over 130 milliseconds, respectively. Percentile analysis is more effective for sporadic peaks and end events in industrial data [27].

Figure 3(c) shows the maximum and minimum latency values found in 25 independent experiments. The minimum latency for the adaptive and static edge methods is approximately 9 milliseconds, but their maximum latency under peak load differs—cloud solutions exceed 150 milliseconds, while the proposed algorithm still maintains a good variance control of 53 milliseconds. Figure 3(d) depicts the delay probability distribution of all scheduling methods. The resource-aware scheduler has a peak unimodal distribution centered around 17

milliseconds, while the static method has a relatively flat right-skewed curve, making it more unpredictable in terms of time guaranties [28].

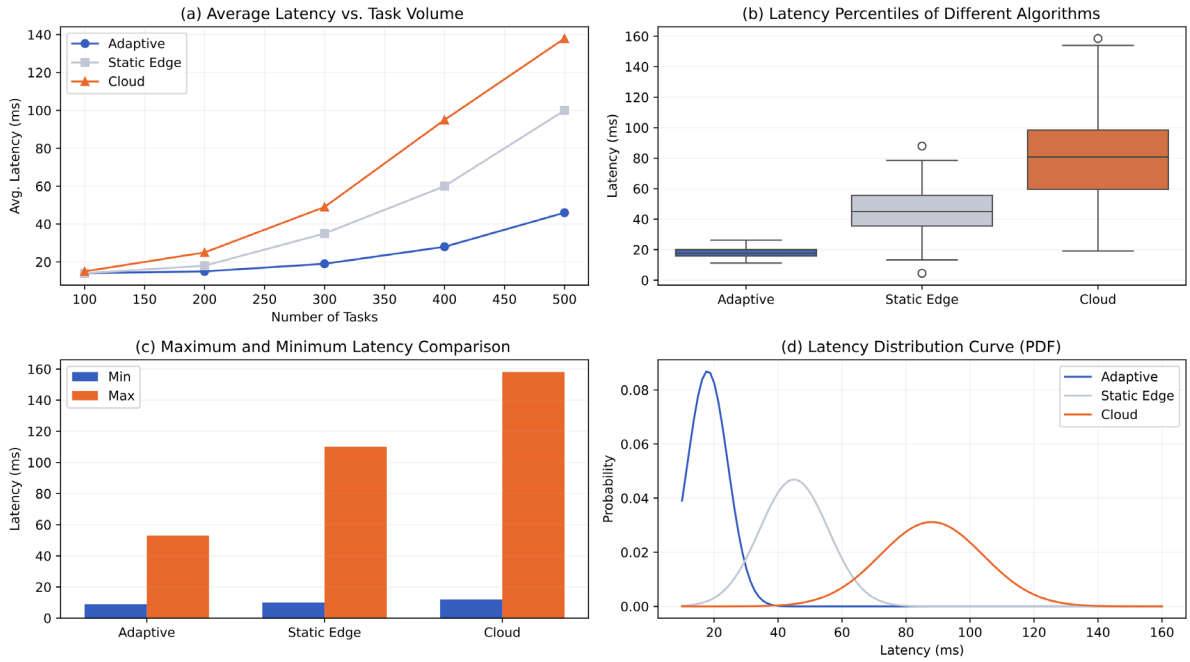

Figure 3. Latency Performance Under Varying Workloads: (a) Average Latency vs. Task Volume; (b) Latency Percentiles of Different Algorithms; (c) Maximum and Minimum Latency Comparison; (d) Latency Distribution Curve.

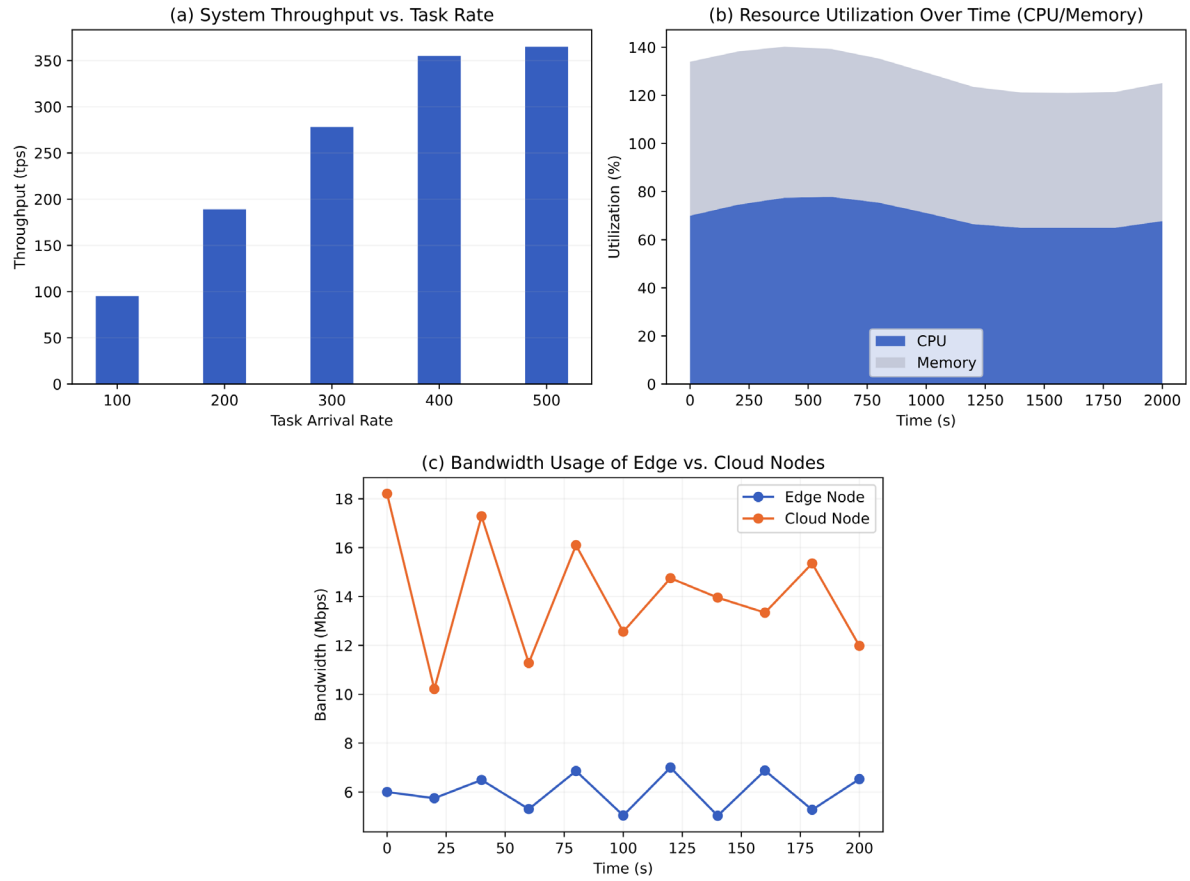

Figure 4. System Throughput and Resource Utilization: (a) System Throughput vs. Task Rate; (b) Resource Utilization Over Time (CPU/Memory); (c) Bandwidth Usage of Edge vs. Cloud Nodes.

The results driven by latency are based on the response speed and stability under different loads. Many researchers have recently studied the evaluation of the impact of improved task volume, allocation strategies, and operating environments on the overall system output and resource consumption. Figure 4(a) shows the relationship between system throughput and task arrival rate. Adaptively adjusts the nearly linear throughput to 470 tasks per second. When the throughput reaches the saturation value, the throughput gradually stabilizes. Under burst loads, both comparators have earlier saturation points, and the throughput is unstable [29].

Figure 4(b) shows the CPU and memory resource utilization of the three algorithms in the first three hours. The new method does not exhibit the huge, unstable peaks in peak load performance shown by the static model; it maintains CPU utilization between 75%-80%, with smaller fluctuations and stable memory usage. Figure 4(c) shows the differences in bandwidth consumption between local processing edge optimization and cloud offloading design. Without affecting the service level, the adaptive method reduces the average data transmission from edge to cloud by 46%. As shown in Figure 4, the bandwidth of the edge nodes is relatively stable, and the peak usage has also significantly decreased [30].

Comparative Studies

Figure 5(a) shows the overall success rates of five common scheduling algorithms: adaptive, static, cloud, hybrid, and random, as the workload complexity increases. At the highest task input rate, the adaptive scheme can still maintain a success rate of over 98%. Static, cloud, hybrid, and random strategies perform poorly under pressure, with the random method dropping to 66% under peak pressure. These algorithms include fixed, cloud-centered, hybrid, and random allocation, each capable of adapting to complex business environments [31].

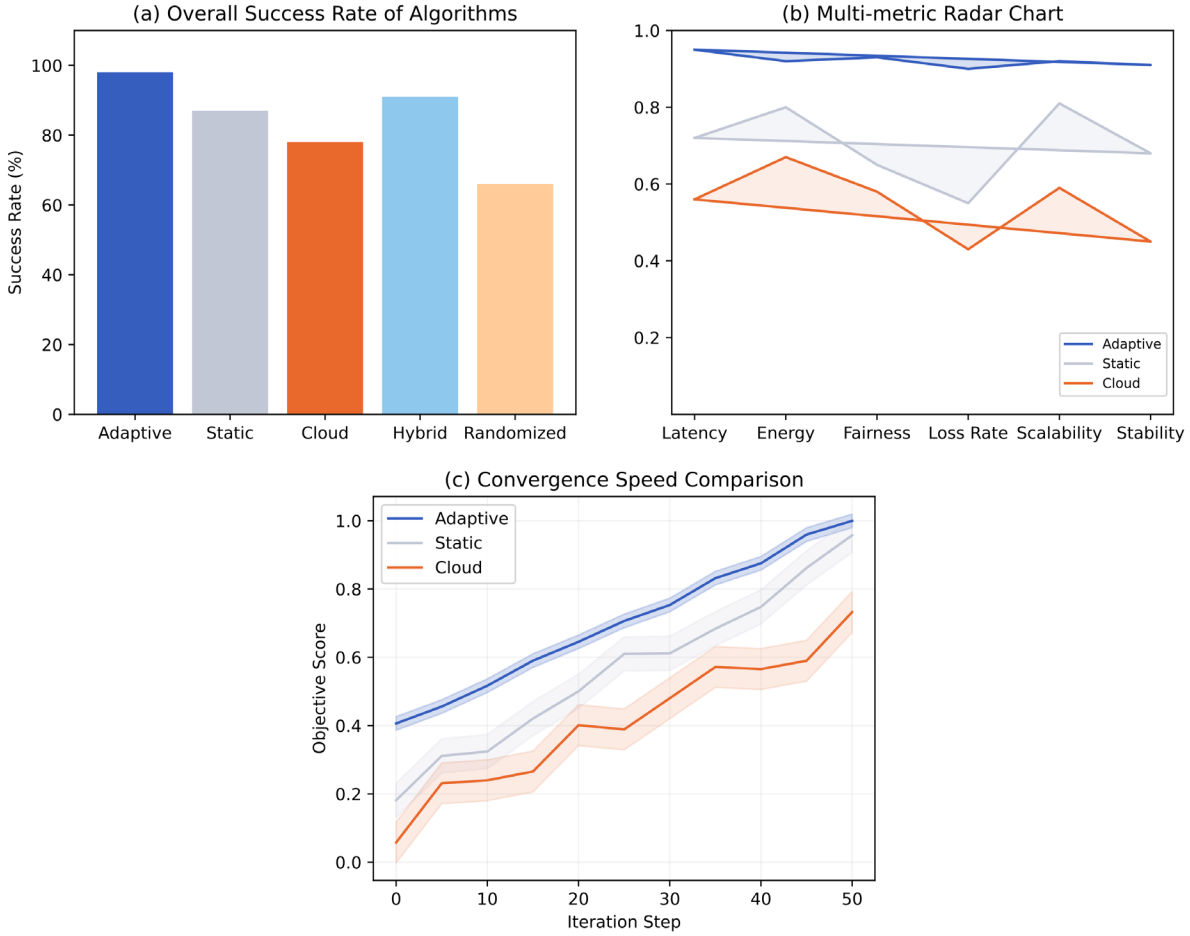

Figure 5. Comparative Analysis of Scheduling Algorithms: (a) Overall Success Rate of Five Algorithms; (b) Multi-metric Radar Chart; (c) Convergence Speed Comparison.

Figure 5(b) shows six main performance metrics, including latency, energy efficiency, fairness, packet loss rate, scalability, and stability. The hybrid method balances static and dynamic decision logic, still being the largest in

terms of area; the adaptive method can utilize intelligent scheduling and robust resource allocation. Cloud and randomized strategies focus on lower levels, exhibiting systemic flaws in delay handling and load fairness. Figure 5(c) is an extended graph, showing the convergence speed and error at least eleven times after the iteration begins. After forty cycles, the adaptive method is relatively stable. Compared to other methods, it performs quite well under increased load or topology changes. The multiple comparisons of various data indicate that the new algorithm not only performs well on individual metrics but also provides reliable and reproducible system-level improvements for high-dimensional, multivariable industrial automation workloads [32].

System Scalability Assessment

Figure 6(a) shows the variation in response time related to the number of edge nodes. The recommended median response time for the scheduler between 4 to 32 nodes is 19 milliseconds and 29 milliseconds. It has high parallelism, and after 32 nodes, the increase in response time is relatively slow. The response penalty for static methods is relatively small, averaging 59 milliseconds at 40 nodes. Therefore, their scalability is lower [33]. Figure 6(b) shows the load distribution of edge nodes under time-varying and heterogeneous inputs. In the static scheme, there are nodes with excessive loads, with peaks three times the average load, while the adaptive scheduler achieves a distribution close to normal, with a standard deviation of less than 8%. This leads to the risk of resource waste and the formation of local bottlenecks.

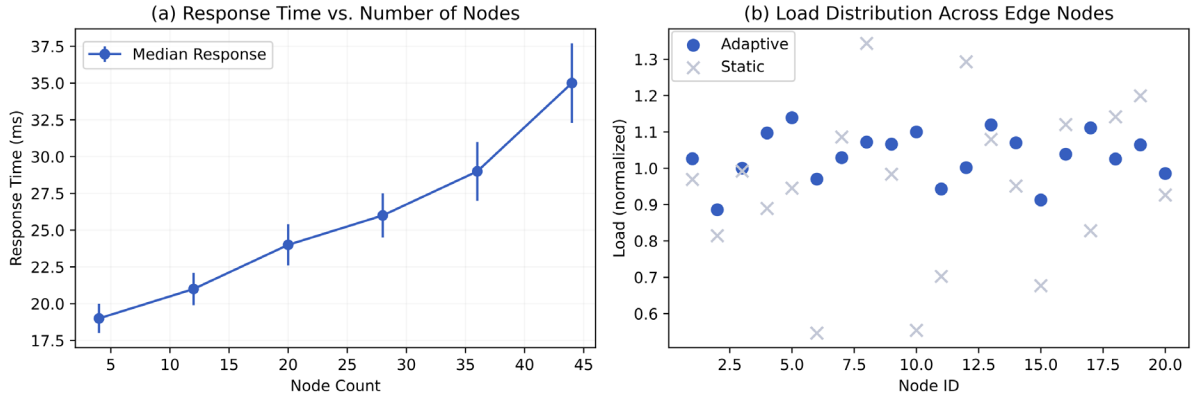

Figure 6. System Scalability Under Heterogeneous Loads: (a) Response Time vs. Number of Nodes; (b) Load Distribution Across Edge Nodes.

The system must have fault tolerance capabilities while being able to handle large-scale loads. Therefore, the robustness and recovery capabilities of the research system will be examined under abnormal events, power outages, and sudden increases in traffic [34]. For robustness analysis, Figure 7(a) shows the distribution of task completion times after random node failures. The adaptive scheduling method demonstrates good recovery capability, with a median completion time of less than 37 milliseconds; the static and alternative methods exhibit heavier tails and some anomalous slowdowns when sampled over multiple failure runs.

Figure 7(b) shows the success rate heatmap of the adaptive, static, and cloud algorithms under different network packet loss rates (0%-12%). Compared to the comparative methods, the adaptive method performs better; its performance decline curve is smoother and less pronounced than that of the comparative methods. Figure 7(c) shows the area chart of CPU and memory resource consumption under normal and peak conditions. This indicates that the adaptive solution limits resource peaks and maintains the predictability of system overhead when the workload increases.

In previous experiments, Figure 7(d) added five different scheduling algorithms (randomized, adaptive, static, cloud, hybrid) to the recovery time after fault injection. The adaptive solution recovers to baseline in an average of 2.6 seconds, while the variance and recovery times of other solutions have increased, with some randomized category solutions exceeding 9 seconds in poor runs. According to the above comprehensive visualization, the proposed scheduling framework has resource efficiency, low latency, and rapid self-healing capabilities. It exhibits good resilience in various algorithms and event scenarios, making it suitable for industrial applications [35].

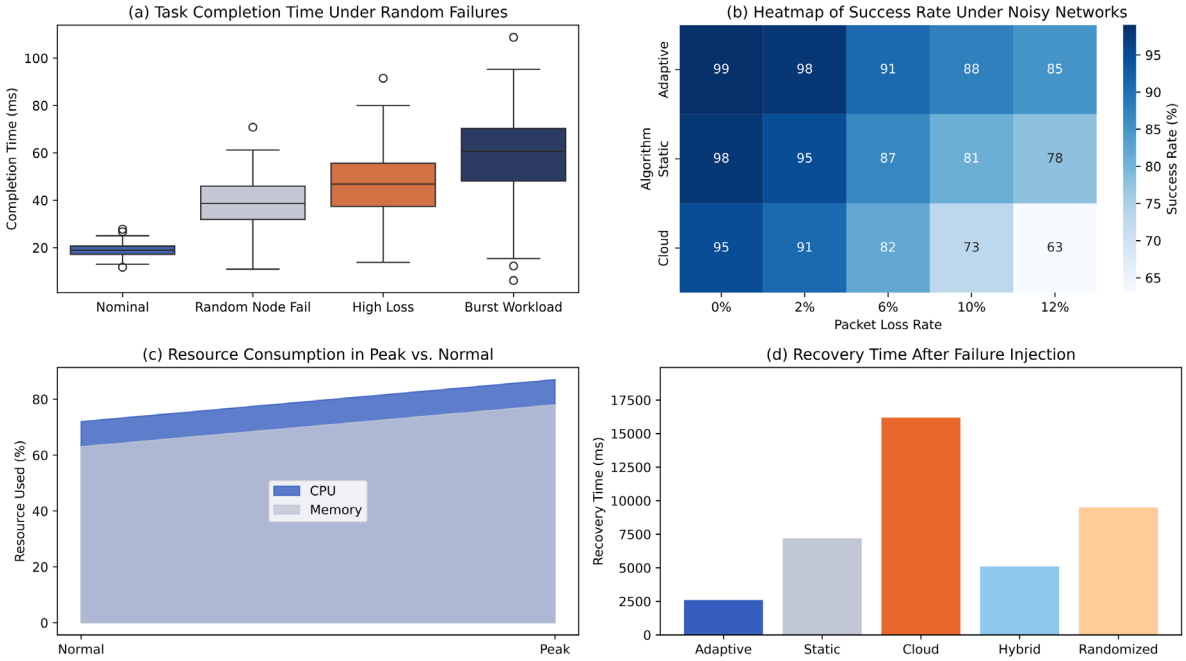

Figure 7. Robustness Analysis in Adverse Scenarios: (a) Task Completion Time Under Random Failures; (b) Heatmap of Success Rate Under Noisy Networks; (c) Resource Consumption in Peak vs. Normal Conditions; (d) Recovery Time After Failure Injection.

Conclusion

To meet the high demands for real-time operation, stability, and scalability in large-scale IIoT environments in industrial automation, this paper proposes an improved edge computing framework. The core of this new system is an adaptive dynamic task offloading and predictive resource-aware scheduling algorithm based on a single modular multi-layer architecture. Clearly model the various capabilities of devices and random workloads to more effectively allocate edge computing resources. This helps achieve ultra-low latency, energy efficiency, and fair system operation under conditions of high variability and operational uncertainty.

In order to verify the actual performance of the system, many experiments were conducted on an industrial-grade testing platform. The experiments show that the edge-enabled architecture avoids system congestion, maintaining an end-to-end latency of less than 20 milliseconds under both normal and peak load conditions. It outperforms static edge and centralized cloud solutions in terms of throughput, recovery speed, and resource utilization. The framework has good network fault tolerance and resilience, with fast recovery speed and minimal performance degradation under stress. Based on experiments with various algorithms and settings, this method can be used to develop mission-critical engineering systems.

In summary, this research provides strong theoretical and practical support for the development of real-time edge intelligence in digital industry and smart manufacturing environments. It not only improves the speed and stability of current solutions but also lays the foundation for the next generation of autonomous and context-aware industrial platforms. In the future, researchers will study federated learning integrated at the edge, deeper context-aware scheduling, and real-time digital twin synchronization to enhance the reliability, flexibility, and intelligence of industrial edge computing in more interconnected and dynamic production environments.

Author Contributions

Grzegorz Gnat contributes to conceptualization, methodology, software, validation, analysis, investigation, data collection, draft preparation, manuscript editing, visualization, supervision. Urszula Fąfara contributes to methodology, software, validation, analysis. All authors have read and agreed with the manuscript before its submission and publication.

Funding

This research received no specific financial support from any funding agency.

Institutional Review Board Statement

Not applicable.

References

[1] Hu, Y., Jia, Q., Yao, Y., Lee, Y., Lee, M., Wang, C., ... & Yu, F. R. (2024). Industrial internet of things intelligence empowering smart manufacturing: A literature review. IEEE Internet of Things Journal, 11(11), 19143-19167. https://doi.org/10.1109/JIOT.2024.3367692

[2] Joha, M. I., Rahman, M. M., Nazim, M. S., & Jang, Y. M. (2024). A secure IIoT environment that integrates AI-driven real-time short-term active and reactive load forecasting with anomaly detection: a real-world application. Sensors, 24(23), 7440. https://doi.org/10.3390/s24237440

[3] Lu, Y., Yang, L., Yang, S. X., Hua, Q., Sangaiah, A. K., Guo, T., & Yu, K. (2022). An intelligent deterministic scheduling method for ultralow latency communication in edge enabled industrial internet of things. IEEE Transactions on Industrial Informatics, 19(2), 1756-1767. https://doi.org/10.1109/TII.2022.3186891

[4] Walia, G. K., Kumar, M., & Gill, S. S. (2023). AI-empowered fog/edge resource management for IoT applications: A comprehensive review, research challenges, and future perspectives. IEEE Communications Surveys & Tutorials, 26(1), 619-669. https://doi.org/10.1109/COMST.2023.3338015

[5] Dong, J., Li, Z., Zheng, Y., Luo, J., Zhang, M., & Yang, X. (2024). Real-time fault detection for IIoT facilities using GA-Att-LSTM based on edge-cloud collaboration. Frontiers in Neurorobotics, 18, 1499703. https://doi.org/10.3389/fnbot.2024.1499703

[6] Mourtzis, D. (2022). Advances in adaptive scheduling in industry 4.0. Frontiers in manufacturing technology, 2, 937889. https://doi.org/10.3389/fmtec.2022.937889

[7] Umer, A., Ali, M., Jehangiri, A. I., Bilal, M., & Shuja, J. (2024). Multi-objective task-aware offloading and scheduling framework for internet of things logistics. Sensors, 24(8), 2381. https://doi.org/10.3390/s24082381

[8] Liso, A., Cardellicchio, A., Patruno, C., Nitti, M., Ardino, P., Stella, E., & Renò, V. (2024). A review of deep learning-based anomaly detection strategies in industry 4.0 focused on application fields, sensing equipment, and algorithms. IEEE Access, 12, 93911-93923. https://doi.org/10.1109/ACCESS.2024.3424488

[9] Chandrika, P. K., Mekala, M. S., & Srivastava, G. (2023). Edge resource slicing approaches for latency optimization in AI-edge orchestration. Cluster Computing, 26(2), 1659-1683. https://doi.org/10.1007/s10586-022-03817-7

[10] Shang, S., Li, X., Gu, K., Li, L., Zhang, X., & Pandi, V. (2023). A robust privacy-preserving data aggregation scheme for edge-supported IIoT. IEEE Transactions on Industrial Informatics, 20(3), 4305-4316. https://doi.org/10.1109/TII.2023.3315375

[11] Tang, X., Zhang, H., Zhang, R., Zhou, D., Zhang, Y., & Han, Z. (2023). Robust trajectory and offloading for energy-efficient UAV edge computing in industrial Internet of Things. IEEE Transactions on Industrial Informatics, 20(1), 38-49. https://doi.org/10.1109/TII.2023.3256375

[12] Farahani, B., & Monsefi, A. K. (2023). Smart and collaborative industrial IoT: A federated learning and data space approach. Digital Communications and Networks, 9(2), 436-447. https://doi.org/10.1016/j.dcan.2023.01.022

[13] Pathak, V., Singh, K., Khan, T., Shariq, M., Chaudhry, S. A., & Das, A. K. (2024). A secure and lightweight trust evaluation model for enhancing decision-making in resource-constrained industrial WSNs. Scientific reports, 14(1), 28162. https://doi.org/10.1038/s41598-024-75414-0

[14] Wang, H., Bai, Y., & Xie, X. (2023). Deep reinforcement learning based resource allocation in delay-tolerance-aware 5G industrial IoT systems. IEEE Transactions on Communications, 72(1), 209-221. https://doi.org/10.1109/TCOMM.2023.3322736

[15] Nguyen, Q. T., Tran, T. N., Heuchenne, C., & Tran, K. P. (2022). Decision support systems for anomaly detection with the applications in smart manufacturing: a survey and perspective. In Machine Learning and Probabilistic Graphical Models for Decision Support Systems (pp. 34-61). CRC Press. https://doi.org/10.1201/9781003189886

[16] Gong, Y., Yao, H., Wang, J., Li, M., & Guo, S. (2022). Edge intelligence-driven joint offloading and resource allocation for future 6G industrial Internet of Things. IEEE Transactions on Network Science and Engineering, 11(6), 5644-5655. https://doi.org/10.1109/TNSE.2022.3141728

[17] Sharma, M., Tomar, A., & Hazra, A. (2024). Edge computing for industry 5.0: Fundamental, applications, and research challenges. IEEE Internet of Things Journal, 11(11), 19070-19093. https://doi.org/10.1109/JIOT.2024.3359297

[18] Ahmed, I. (2024). Deploying Low-Latency Edge AI in Medical IOT Networks: A Case Study of Secure Real-Time Patient Monitoring Systems. American Journal of Scholarly Research and Innovation, 3(02), 337-374. https://doi.org/10.63125/x8255a80

[19] Wang, W., Hu, T., & Gu, J. (2022). Edge-cloud cooperation driven self-adaptive exception control method for the smart factory. Advanced Engineering Informatics, 51, 101493. https://doi.org/10.1016/j.aei.2021.101493

[20] Ai, L., Tan, B., Zhang, J., Wang, R., & Wu, J. (2022). Dynamic offloading strategy for delay-sensitive task in mobile-edge computing networks. IEEE Internet of Things Journal, 10(1), 526-538. https://doi.org/10.1109/JIOT.2022.3202797

[21] Yin, Y., Wang, L., Hoang, D. T., Wang, W., & Niyato, D. (2024). Sparse attention-driven quality prediction for production process optimization in digital twins. IEEE Internet of Things Journal, 11(23), 38569-38584. https://doi.org/10.1109/JIOT.2024.3448256

[22] Ramzey, H., Badawy, M., Elhosseini, M., & A. Elbaset, A. (2023). I2OT-EC: A framework for smart real-time monitoring and controlling crude oil production exploiting IIOT and edge computing. Energies, 16(4), 2023. https://doi.org/10.3390/en16042023

[23] Mustafa, R., Sarkar, N. I., Mohaghegh, M., & Pervez, S. (2024). A cross-layer secure and energy-efficient framework for the internet of things: A comprehensive survey. Sensors, 24(22), 7209. https://doi.org/10.3390/s24227209

[24] Yan, C., Xia, Y., Yang, H., & Zhan, Y. (2024). Cloud control for IIoT in a cloud-edge environment. Journal of Systems Engineering and Electronics, 35(4), 1013-1027. https://doi.org/10.23919/JSEE.2024.000074

[25] Al-Dulaimy, A., Ashjaei, M., Behnam, M., Nolte, T., & Papadopoulos, A. V. (2022, November). Fault tolerance in cloud manufacturing: An overview. In International Conference on Mobile Computing, Applications, and Services (pp. 89-101). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-31891-7_7

[26] Lyu, L., Zhao, L., Dai, Y., Cheng, N., Chen, C., Guan, X., & Shen, X. (2022). Adaptive edge sensing for industrial IoT systems: Estimation task offloading and sensor scheduling. IEEE Internet of Things Journal, 10(1), 391-402. https://doi.org/10.1109/JIOT.2022.3200392

[27] Zhao, M., Zhang, X., He, Z., Chen, Y., & Zhang, Y. (2024). Dependency-aware task scheduling and layer loading for mobile edge computing networks. IEEE Internet of Things Journal, 11(21), 34364-34381. https://doi.org/10.1109/JIOT.2024.3382682

[28] Liu, H., Li, S., Li, W., & Sun, W. (2023). A distributed resource sharing mechanism in edge-enabled IIoT systems. IEEE Internet of Things Journal, 11(8), 14296-14312. https://doi.org/10.1109/JIOT.2023.3342321

[29] Peng, Y., Jolfaei, A., Hua, Q., Shang, W. L., & Yu, K. (2022). Real-time transmission optimization for edge computing in industrial cyber-physical systems. IEEE Transactions on Industrial Informatics, 18(12), 9292-9301. https://doi.org/10.1109/TII.2022.3181199

[30] Rashidi, S., Won, W., Srinivasan, S., Sridharan, S., & Krishna, T. (2022, June). Themis: A network bandwidth-aware collective scheduling policy for distributed training of dl models. In Proceedings of the 49th Annual International Symposium on Computer Architecture (pp. 581-596). https://doi.org/10.1145/3470496.3527382

[31] Yin, H., Huang, X., & Cao, E. (2024). A cloud-edge-based multi-objective task scheduling approach for smart manufacturing lines. Journal of Grid Computing, 22(1), 9. https://doi.org/10.1007/s10723-023-09723-5

[32] Guo, Z., Lu, Y., Tian, H., Zuo, J., & Lu, H. (2023). A security evaluation model for multi-source heterogeneous systems based on IOT and edge computing. Cluster Computing, 26(1), 303-317. https://doi.org/10.1007/s10586-021-03410-4

[33] Harjula, E., Artemenko, A., & Forsström, S. (2020). Edge computing for industrial IoT: challenges and solutions. In Wireless Networks and industrial IoT: applications, challenges and enablers (pp. 225-240). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-51473-0_12

[34] Zhou, Q., Zhao, T., Chen, X., Zhong, Y., & Luo, H. (2022). A fault-tolerant transmission scheme in SDN-based industrial IoT (IIoT) over fiber-wireless networks. Entropy, 24(2), 157. https://doi.org/10.3390/e24020157

[35] Adeniyi, O., Sadiq, A. S., Pillai, P., Taheir, M. A., & Kaiwartya, O. (2023). Proactive self-healing approaches in mobile edge computing: A systematic literature review. Computers, 12(3), 63. https://doi.org/10.3390/computers12030063