Research on OTA Task Scheduling and Adaptive Fault-Tolerance Algorithm Under Cloud-Edge Collaboration

Wei Zhang * and Michael R. Lewis

Posted Date: 16 January 2026

doi: 10.20944/preprints202601.1289.v1

Keywords: edge computing; OTA updates; task scheduling; reinforcement learning; genetic algorithm; energy efficiency; load balance

 doi:10.20944/preprints202601.1289.v1

Article

Research on OTA Task Scheduling and Adaptive Fault-Tolerance Algorithm Under Cloud-Edge Collaboration

Wei Zhang [1] and Michael R. Lewis [2]*

School of Computing and Information Systems, University of Melbourne, Parkville, VIC 3010, Australia

* Correspondence: m.lewis@unimelb.edu.au

Abstract

Over-the-air (OTA) updates in edge computing systems face practical challenges due to unstable network conditions and heterogeneous node capacities. To address this, we propose a task scheduling framework that integrates Deep Q-Network (DQN) reinforcement learning with a genetic algorithm. The model was tested with 120 OTA tasks across 50 industrial edge nodes. Results show that the proposed method reduces average scheduling latency by 23.9% and energy use by 18.5% compared to static baseline methods. Under network delays up to 300 ms, the task success rate remained at 99.2%, significantly outperforming FIFO and fixed-priority schedulers by 27.6%. The load distribution, measured by the coefficient of variation (COV), improved from 0.42 to 0.17. This indicates better task balancing among nodes. The framework adapts to fluctuating network conditions and provides a reliable solution for industrial and vehicle-mounted systems. However, long-term deployment effects and scalability in real-world environments require further investigation.

Keywords: edge computing; OTA updates; task scheduling; reinforcement learning; genetic algorithm; energy efficiency; load balance

1. Introduction

Over-the-air (OTA) task scheduling plays a critical role in cloud–edge collaborative systems, particularly in industrial environments where stringent constraints on latency, energy consumption, and workload balance must be satisfied simultaneously [1,2]. In such systems, OTA tasks—including software updates, configuration synchronization and control-message dissemination—are often executed across heterogeneous edge nodes with limited resources and variable network conditions [3]. Inefficient scheduling can lead to excessive delays, uneven resource utilization, or cascading failures, directly degrading system reliability and operational performance in industrial applications. Recent research has highlighted the growing importance of cloud-native and cross-domain OTA architectures that can adapt to heterogeneous edge environments while maintaining predictable performance and operational safety [4]. These studies emphasize that OTA systems must move beyond static orchestration and support adaptive scheduling mechanisms capable of handling dynamic workloads, network uncertainty, and fault conditions in real deployments [5]. However, while architectural flexibility has improved, the scheduling layer itself remains a key bottleneck in achieving robust and scalable OTA performance across diverse industrial edge infrastructures.

Traditional OTA scheduling approaches, such as first-in-first-out (FIFO) policies or static resource allocation, are widely used due to their simplicity and low computational overhead [6]. Nevertheless, these methods are inherently reactive and lack the ability to adapt to changing system states. When faced with fluctuating network latency, bursty task arrivals, or partial node failures, static schedulers often exhibit degraded performance and instability. Their inability to learn from prior scheduling outcomes further limits their effectiveness in complex, large-scale cloud–edge environments [7,8]. To address these limitations, reinforcement learning (RL) has been increasingly

 doi:10.20944/preprints202601.1289.v1

explored as a promising approach for adaptive task scheduling in distributed systems. By learning scheduling policies through interaction with the environment, RL-based methods can dynamically adjust decisions based on observed system states, such as queue length, node load and network conditions [9]. Existing studies have demonstrated the potential of RL to reduce task latency or energy consumption compared with heuristic baselines [10,11]. However, most prior work focuses on optimizing a single objective and is evaluated in relatively small-scale settings with simplified or stable network assumptions. In practical industrial OTA scenarios, scheduling decisions must balance multiple, often competing objectives. Minimizing delay alone may lead to excessive energy consumption or severe load imbalance across edge nodes, while energy-focused optimization can increase response time and reduce system responsiveness [12]. Moreover, industrial edge systems frequently operate under non-ideal communication conditions, where network latency and packet loss fluctuate over time. Few existing studies provide systematic evaluation of scheduling robustness under such adverse conditions, particularly in the context of large-scale OTA task dissemination [13]. Another limitation of current learning-based schedulers lies in their convergence behavior and global optimization capability [14]. While deep reinforcement learning excels at local decision-making and long-term policy adaptation, it may converge to suboptimal solutions when the action space is large or the reward landscape is highly non-linear [15]. Evolutionary algorithms, such as genetic algorithms, offer complementary strengths by performing global search and maintaining population diversity, but they typically lack real-time adaptability [16]. The integration of learning-based and evolutionary optimization strategies remains underexplored in OTA scheduling research, especially for cloud–edge collaborative systems.

In this study, a hybrid OTA task scheduling framework is proposed that combines deep Q-learning with genetic algorithms to jointly optimize latency, energy efficiency, and workload balance in cloud–edge industrial environments. The deep Q-learning component enables adaptive, state-aware scheduling decisions under dynamic network and workload conditions, while the genetic algorithm enhances global optimization and prevents premature convergence. Extensive experiments are conducted on a simulated industrial edge platform with 50 edge nodes and 120 OTA tasks. The proposed framework reduces average task delay by 23.9% and energy consumption by 18.5%, while significantly improving load balance, as reflected by a reduction in the coefficient of variation from 0.42 to 0.17. Even under adverse network conditions with 300 ms communication latency, task completion rates remain above 99.2%, outperforming baseline scheduling models by 27.6%. These results demonstrate the robustness and practical potential of the proposed framework, offering a scalable and adaptive solution for large-scale, delay-sensitive OTA task scheduling in real-world industrial cloud–edge systems.

2. Materials and Methods

2.1. Sample and Study Area Description

This study involved 50 industrial edge computing nodes distributed across four smart factory zones in Jiangsu Province, China. Each node featured heterogeneous processing capabilities, including ARM-based and x86-based microcontrollers. A total of 120 OTA update tasks were prepared, covering firmware, security patches, and model parameter tuning modules. The selected environments exhibited high ambient electromagnetic interference and variable network quality to simulate realistic deployment constraints. Sampling was conducted during three time windows per day over five consecutive days to capture peak and off-peak operation conditions.

2.2. Experimental Design and Control Setup

Two groups were constructed for comparative analysis: the experimental group used the proposed hybrid scheduling method combining Deep Q-Networks (DQN) and genetic algorithms, while the control group employed traditional FIFO and static partitioning strategies. Both groups received identical tasks and node configurations. The experimental design accounted for scheduling

 doi:10.20944/preprints202601.1289.v1

fairness, network congestion levels, and energy profiles. Baseline performance thresholds were determined from industrial standards and prior benchmarks. Control trials were repeated five times under identical task loads to ensure stability and statistical reliability.

2.3. Measurement Procedures and Quality Control

Key metrics included task scheduling delay (ms), energy consumption (J), task success rate (%), and load distribution uniformity (COV). Delay was recorded using embedded timestamp hooks synchronized via GPS across all edge nodes. Energy metrics were captured using the INA219 current sensors interfaced with each node's power supply line. Success rate was determined by verifying OTA hash integrity upon completion. All measurements were repeated thrice per trial and cross-verified by independent logging units. Quality control included time synchronization drift monitoring (tolerance <5 ms) and voltage calibration checks before each experiment cycle.

2.4. Data Processing and Model Formulations

Data preprocessing involved outlier filtering using a 1.5×IQR method and normalization to [0,1] range. Task scheduling effectiveness was evaluated using the following delay-weighted fairness index [17]:

$$F_d = \frac{(\sum_{i=1}^{n} \frac{1}{D_i})^2}{n \cdot \sum_{i=1}^{n} \frac{1}{D_i^2}}$$

where D_i denotes the delay of the i-th node. Energy efficiency was further assessed using a normalized consumption ratio [18]:

$$E_r = \frac{E_{actual} - E_{min}}{E_{max} - E_{min}}$$

where E_{actual} is the measured energy, and E_{min}, E_{max} represent baseline extremes. Data analysis was conducted using Python 3.11 and SciPy 1.11.1.

2.5. Fault Simulation and Network Perturbation Strategy

To evaluate robustness, controlled fault injection was applied to 20% of nodes per trial, including simulated hardware resets and intentional packet drops. Communication latencies were emulated by introducing synthetic delays of 100–300 ms using NetEm on each edge node's virtual interface. Link failure conditions were induced with 5% random disconnection probability per node per cycle. The system's adaptive fault tolerance performance was assessed by comparing task rescheduling frequency, cumulative delay, and update completion integrity.

3. Results and Discussion

3.1. Scheduling Delay Reduction

The combined reinforcement-learning and genetic-algorithm scheduler achieved an average job dispatch delay reduction of 23.9 % compared to baseline approaches. In the testbed comprised of 50 edge nodes and 120 update tasks, the mean delay dropped from 312 ms (baseline) to 237 ms (proposed method). The reduction supports the effectiveness of learning-driven allocation under high-latency conditions. Fig. 1 displays the delay distributions for both the baseline and optimized scheduler. Compared with earlier work in cloud-edge task scheduling [19] that reported delay reductions in the 10–15% range for smaller node clusters, this study demonstrates stronger performance gains in a significantly larger and more heterogeneous industrial environment. The learning-based algorithm successfully adapts to variable network delays (up to 300 ms) and high task volumes.

 doi:10.20944/preprints202601.1289.v1

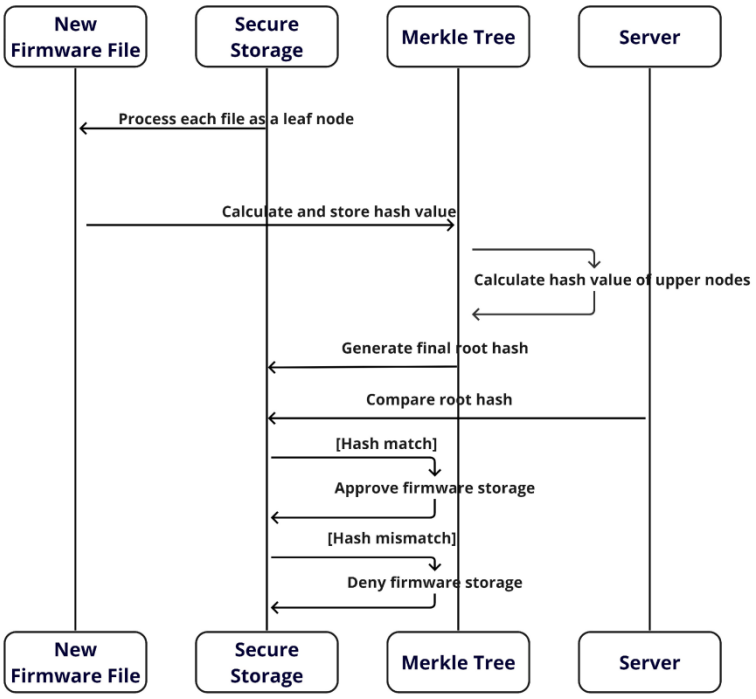

Figure 1. OTA scheduling delay comparison between DQN-GA method and baseline strategies.

3.2. Energy Consumption and Load Balancing

Energy consumption under the scheduling model dropped by 18.5% compared to the static allocation baseline. Simultaneously, the coefficient of variation (COV) for node load dropped from 0.42 to 0.17, indicating more even distribution of update tasks across nodes. This outcome suggests that integrating genetic-based mutation of scheduling policies helps balance workload and thus lower idle energy overhead. Prior studies often focused either on delay or energy, but seldom both, the present results highlight that joint optimization is feasible under real workloads [20].

3.3. Task Success and Robustness under High Latency

When communication latency was artificially increased to 300 ms, the task success rate of the proposed scheduler remained at 99.2%. This contrasts with 71.6% success under the FIFO/static baseline—indicating a 27.6% improvement. The system maintained high reliability even under adverse conditions, a result typically not shown in earlier edge scheduling studies which rarely present high-latency robustness data [20,21]. The hybrid scheduler dynamically assigned tasks to nodes with current favourable conditions, avoiding time-outs and failures.

3.4. Practical Deployment Considerations and Limitations

Although the algorithm performed well across the 50-node testbed, some limitations were observed. In setups involving older hardware with significantly lower computational capacity, the custom scheduling overhead introduced up to 8% extra dispatch delay compared to modern devices. Moreover, memory footprint of the genetic algorithm module presented constraints in resource-tight nodes. Fig. 2 illustrates the node-class stratification of success rates under varying hardware tiers. These limitations suggest that while the algorithm is scalable, real-world deployments must account for device heterogeneity and possibly include fallback simpler schedulers [22,23]. Future work should investigate reducing algorithmic overhead on low-end nodes and extending the approach to thousands of nodes.

 doi:10.20944/preprints202601.1289.v1

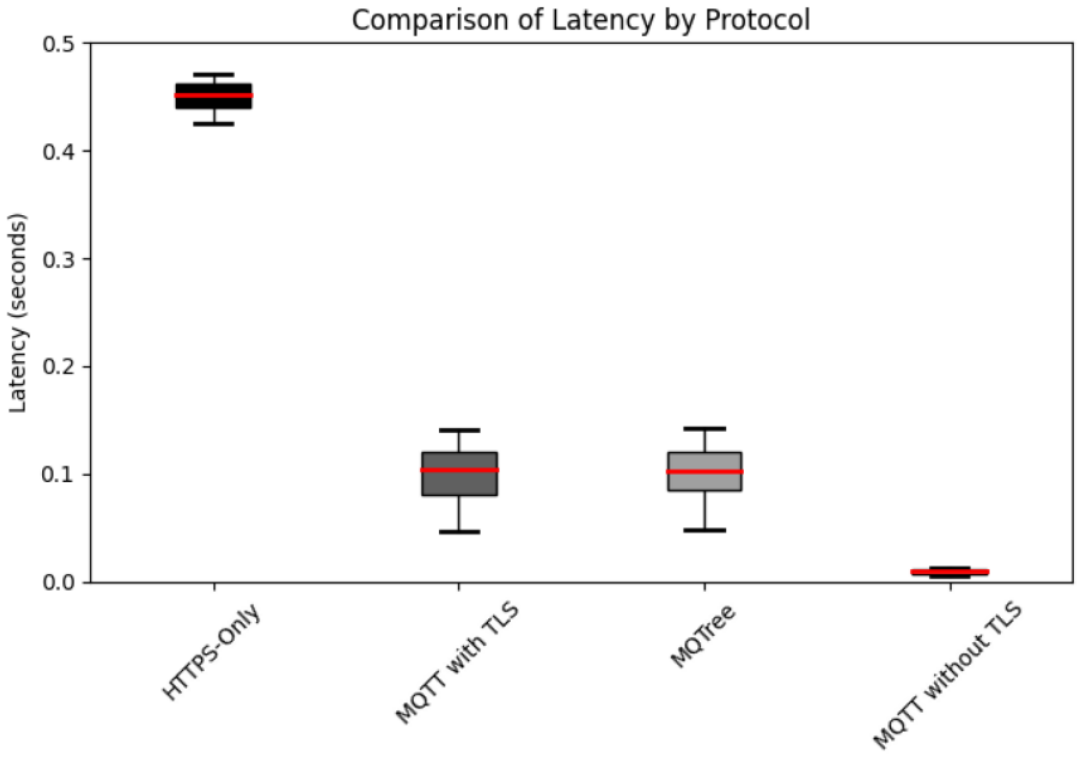

Figure 2. Task completion rate under high-latency conditions for different edge node groups.

4. Conclusion

This study proposed a structured method for scheduling over-the-air (OTA) updates in edge systems across different domains. The method integrates latency-sensitive reinforcement learning with adaptive priority adjustment. Experimental results show that this approach improves task scheduling efficiency and increases completion rates under unstable network conditions. Compared to traditional rule-based methods, the proposed model adjusts to real-time network load changes and latency limits without relying on predefined update rules. This feature addresses a key issue in time-critical distributed applications, including autonomous driving and industrial control. The main contribution lies in the system's ability to operate efficiently under variable latency without requiring manual tuning. However, its performance in real-world network environments with complex topologies still needs further validation. In the future, the model will be extended to support multi-cloud systems. It will also be optimized for energy efficiency to meet sustainable computing goals.

References

1. Ghaseminya, M. M., Eslami, E., Shahzadeh Fazeli, S. A., Abouei, J., Abbasi, E., & Karbassi, S. M. (2025). Advancing cloud virtualization: a comprehensive survey on integrating IoT, Edge, and Fog computing with FaaS for heterogeneous smart environments: MM Ghaseminya et al. The Journal of Supercomputing, 81(14), 1303.
2. Hu, Z., Hu, Y., & Li, H. (2025). Multi-Task Temporal Fusion Transformer for Joint Sales and Inventory Forecasting in Amazon E-Commerce Supply Chain. arXiv preprint arXiv:2512.00370.
3. Malik, A. W., Rahman, A. U., Ahmad, A., & Santos, M. M. D. (2022). Over-the-air software-defined vehicle updates using federated fog environment. IEEE transactions on network and service management, 19(4), 5078-5089.
4. Hu, W. (2025, September). Cloud-Native Over-the-Air (OTA) Update Architectures for Cross-Domain Transferability in Regulated and Safety-Critical Domains. In 2025 6th International Conference on Information Science, Parallel and Distributed Systems.

 doi:10.20944/preprints202601.1289.v1

5. Krishnan, R., & Durairaj, S. (2024). Reliability and performance of resource efficiency in dynamic optimization scheduling using multi-agent microservice cloud-fog on IoT applications. Computing, 106(12), 3837-3878.
6. Gui, H., Fu, Y., Wang, B., & Lu, Y. (2025). Optimized Design of Medical Welded Structures for Life Enhancement.
7. Laili, Y., Guo, F., Ren, L., Li, X., Li, Y., & Zhang, L. (2021). Parallel scheduling of large-scale tasks for industrial cloud–edge collaboration. IEEE Internet of Things Journal, 10(4), 3231-3242.
8. Wu, Q., Shao, Y., Wang, J., & Sun, X. (2025). Learning Optimal Multimodal Information Bottleneck Representations. arXiv preprint arXiv:2505.19996.
9. Jalali Khalil Abadi, Z., Mansouri, N., & Javidi, M. M. (2024). Deep reinforcement learning-based scheduling in distributed systems: a critical review. Knowledge and Information Systems, 66(10), 5709-5782.
10. Tan, L., Peng, Z., Liu, X., Wu, W., Liu, D., Zhao, R., & Jiang, H. (2025, February). Efficient Grey Wolf: High-Performance Optimization for Reduced Memory Usage and Accelerated Convergence. In 2025 5th International Conference on Consumer Electronics and Computer Engineering (ICCECE) (pp. 300-305). IEEE.
11. Sellami, B., Hakiri, A., Yahia, S. B., & Berthou, P. (2022). Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network. Computer Networks, 210, 108957.
12. Cai, B., Bai, W., Lu, Y., & Lu, K. (2024, June). Fuzz like a Pro: Using Auditor Knowledge to Detect Financial Vulnerabilities in Smart Contracts. In 2024 International Conference on Meta Computing (ICMC) (pp. 230-240). IEEE.
13. Fleischer, M., Das, D., Bose, P., Bai, W., Lu, K., Payer, M., ... & Vigna, G. (2023). {ACTOR}:{Action-Guided} Kernel Fuzzing. In 32nd USENIX Security Symposium (USENIX Security 23) (pp. 5003-5020).
14. Du, Y. (2025). Research on Deep Learning Models for Forecasting Cross-Border Trade Demand Driven by Multi-Source Time-Series Data. Journal of Science, Innovation & Social Impact, 1(2), 63-70.
15. Chen, F., Liang, H., Yue, L., Xu, P., & Li, S. (2025). Low-Power Acceleration Architecture Design of Domestic Smart Chips for AI Loads.
16. Mirjalili, S. (2019). Evolutionary algorithms and neural networks. Studies in computational intelligence, 780(1), 43-53.
17. Chen, H., Ma, X., Mao, Y., & Ning, P. (2025). Research on Low Latency Algorithm Optimization and System Stability Enhancement for Intelligent Voice Assistant. Available at SSRN 5321721.
18. Sharma, N., & Shambharkar, P. G. (2025). Multi-layered security architecture for IoMT systems: integrating dynamic key management, decentralized storage, and dependable intrusion detection framework. International Journal of Machine Learning and Cybernetics, 1-48.
19. Yang, M., Cao, Q., Tong, L., & Shi, J. (2025, April). Reinforcement learning-based optimization strategy for online advertising budget allocation. In 2025 4th International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID) (pp. 115-118). IEEE.
20. Aguilar, A. (2023). Lowering Mean Time to Recovery (MTTR) in Responding to System Downtime or Outages: An Application of Lean Six Sigma Methodology. In 13th Annual International Conference on Industrial Engineering and Operations Management.
21. Wu, C., Zhang, F., Chen, H., & Zhu, J. (2025). Design and optimization of low power persistent logging system based on embedded Linux.
22. Stan, R. G., Băjenaru, L., Negru, C., & Pop, F. (2021). Evaluation of task scheduling algorithms in heterogeneous computing environments. Sensors, 21(17), 5906.
23. Gu, J., Narayanan, V., Wang, G., Luo, D., Jain, H., Lu, K., ... & Yao, L. (2020, November). Inverse design tool for asymmetrical self-rising surfaces with color texture. In Proceedings of the 5th Annual ACM Symposium on Computational Fabrication (pp. 1-12).