Resource-Efficient Federated Learning and DAG Blockchain With Sharding in Digital-Twin-Driven Industrial IoT

Li Jiang, *Member, IEEE*, Yi Liu, Hui Tian, *Senior Member, IEEE*, Lun Tang, and Shengli Xie, *Fellow, IEEE*

***Abstract*—The development of Industry 4.0 relies on emerging technologies of digital twin, machine learning, blockchain, and Internet of Things (IoT) to build autonomous self-configuring systems that maximize manufactory efficiency, precision, and accuracy. In this article, we propose a new distributed and secure digital twin-driven IIoT framework that integrates federated learning and directed acyclic graph (DAG) blockchain with sharding. The proposed framework includes three planes: 1) the data plane; 2) the blockchain plane; and 3) the digital twin plane. Specifically, the data plane performs federated learning through a set of cluster heads to train models at network edges for twin model construction. The blockchain plane, which supports sharding, utilizes a hierarchical consensus scheme based on DAG blockchain to verify both local model updates and global model updates. The digital twin plane is responsible for constructing and maintaining twin model. Then, an efficient resource scheduling scheme is designed by considering performance of both federated learning and DAG blockchain with sharding. Accordingly, an optimization problem is formulated to maximize long-term utility of the digital twin-driven IIoT. To cope with mapping error in the digital twin plane, a multiagent proximal policy optimization (MAPPO) approach is developed to solve the optimization problem. Numerical results illustrate that comparing with traditional approach, the proposed MAPPO improves utility by about 37 %, and reduces time latency by about 14%. Moreover, it also can well adapt to the mapping error.**

***Index Terms*—Digital twin, directed acyclic graph (DAG) blockchain with sharding, federated learning, Industrial Internet of Things (IIoT), multiagent proximal policy optimization (MAPPO), resource scheduling.**

Manuscript received 24 October 2023; revised 2 January 2024; accepted 14 January 2024. Date of publication 24 January 2024; date of current version 9 May 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1807801 and Grant 2020YFB1807800; in part by the NSFC Programs under Grant 62371142 and Grant 62273107; and in part by the Engineering Research Center of Mobile Communications, Ministry of Education under Grant cqupt-mct-202003. *(Corresponding author: Yi Liu.)*

Li Jiang is with the School of Automation and the Guangdong Province Key Laboratory of IoT Information Technology, Guangdong University of Technology, Guangzhou 510006, China (e-mail: jiangli@gdut.edu.cn).

Yi Liu is with the Guangdong–Hong Kong–Macao Joint Laboratory for Smart Discrete Manufacturing and the Key Laboratory of Intelligent Detection and Internet of Manufacturing Things, Ministry of Education, Guangdong University of Technology, Guangzhou 510006, China (e-mail: yi.liu@gdut.edu.cn).

Hui Tian is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: tianhui@bupt.edu.cn).

Lun Tang is with the School of Communication and Information Engineering and the Key Laboratory of Mobile Communication, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: tangl@cqupt.edu.cn).

Shengli Xie is with the 111 Center for Intelligent Batch Manufacturing Based on IoT Technology and the Key Laboratory of Intelligent Information Processing and System Integration of IoT, Ministry of Education, Guangdong University of Technology, Guangzhou 510006, China (e-mail: shlxie@gdut.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3357827

I. Introduction

INDUSTRIAL Internet of Things (IIoT), which is also known as industrial Internet, has been regarded as a revolutionary approach to improve efficiency of industrial production, especially in the context of Industry 4.0 [1]. Ubiquitous smart devices are deployed in IIoT to monitor and sense ambient environment, enabling real-time data collection. Then, the collected data is sent to nearby access points (APs) for analysis and processing, facilitating intelligent decision making and enhancing the accuracy of manufacturing process. Furthermore, the deployment of digital twin at the factory networks' edge, which incorporates mobile-edge computing (MEC) functions, is considered as a critical scenario in the context of IIoT in the Industry 4.0 era. The digital twin technology fosters the digitalization and intelligentization of industrial processes, while providing a platform for transparent and secure data exchange [2], [3], [4], [5], [6].

The deployed digital twin at the factory networks' edge can form digital twin edge networks, where the edge servers create the twin model locally from the incoming data, such as sensory data, historical data and simulation data, for comprehensive data analysis and obtaining accurate depiction of practical situations. Moreover, the edge servers update the local edge twin model by continuously interacting with the physical entities, so as to keep consistency with the twin mappings. Hence, comparing with traditional cloud-based digital twin, where the centralized server collects and processes the heterogeneous data for twin model creation [7], the digital twin edge networks facilitate flexible twin model construction in IIoT [8], [9], [10], [11]. Applying machine learning to edge twin model construction can bring intelligence to digital twin-based industrial applications [12], [13], [14]. Specifically, the machine learning algorithms deployed in the cloud/edge servers can build performance parameters from raw data, which can be utilized to update the edge twin model. Despite machine learning plays an important role in

edge twin model construction, there are new challenges for deploying machine learning in digital twin edge networks due to security and privacy concerns: 1) it is insecure for physical entities to share the enormous amount of raw data with their edge twin models by considering about privacy and 2) conventional intermediary dependent management of machine learning parameters sharing may suffer scalability issues and single-point-of-failure problem. To this end, state of the art demands new approaches for enhancing security and privacy for twin model construction in digital twin edge networks.

Federated learning is a distributed learning method to alleviate privacy leakage for factory sensitive data in digital twin edge networks [15], where each client device trains local model based on their local data, and uploads local model parameters to master node (e.g., edge server) instead of sending original data. Then, the master node aggregates the collected local model parameters to update the edge twin model. Recently, some works have studied the advanced federated learning in digital twin edge networks to improve communication efficiency and accuracy [16], [17], [18]. However, there are still some security challenges when applying federated learning to construct edge twin model in IIoT. Since the edge server which owns all of the client twin models may mislead the behavior of the physical devices. Besides, malicious client devices may disseminate low-quality model parameters to the edge server to adversely affect the twin model construction. Blockchain is a distributed and decentralized solution to enable secure interactions among untrusted individuals [19], [20], [21], [22], [23]. The fusion of federated learning and blockchain in digital twin edge networks has been studied in [24] and [25]. These studies explore the use of blockchain to record and verify model updates in federated learning, thereby strengthening security and privacy protection for edge twin model construction.

Although the fusion of federated learning and blockchain brings significant benefits for edge twin model construction, the challenges lie in their integration and optimization to work harmoniously in practical digital twin-driven IIoT. *It may take long time to finish model updates confirmation*, since both model parameters training and model updates verification are executed in time-varying and stochastic IIoT wireless environment. On the other hand, the blockchain mainly depends on consensus schemes to verify and confirm the model updates. It is noted that the traditional proof-based consensus schemes, such as Proof of Work and Proof of Stake, suffer from limitations of scalability and cannot meet low-latency requirement for the model updates confirmation. Directed acyclic graph (DAG)-based IOTA ledger [27] has been developed to process a large amount of transactions faster in IoT by depending on cumulative weight-based consensus scheme. Motivated by the layered idea of heterogeneous blockchain [28], DAG blockchain is developed by integrating DAG and consortium blockchain to accommodate the different IoT use cases. In [29], DAG blockchain was designed to secure the wireless power transfer in UAVs participated networks framework. In [9], DAG blockchain was proposed for enabling secure model updates verification in digital twin edge networks. However, the vast and growing scale of transactions cause the DAG blockchain being compromised to the scalability of resource limited physical devices in IIoT. To cope with this issue, sharding technology has been taken to boost blockchain's scalability [30], [31], [32], [33]. In blockchain sharding, the recorded transactions are divided into multiple disjoint subsets which are maintained by different committees. In this way, the high-volume transactions can be processed and recorded concurrently. Nonetheless, there is not explicit framework to utilize federated learning and DAG blockchain with sharding for edge twin model construction in IIoT. Besides, the integrated federated learning and DAG blockchain with sharding incurs resource costs in terms of computing power and communication bandwidth for the resource limited IIoT devices. Moreover, the available resource of the devices is diverse at different time instances, which may lead to failure in model updates confirmation.

Resource scheduling is an efficient method to improve performance of both federated learning and blockchain in IIoT. To adapt to the time-varying and stochastic features of the digital twin-driven IIoT environment, deep RL (DRL) has been increasingly leveraged to solve complex resource scheduling problems, providing optimal scheduling decisions for the scenarios [34]. Specifically, DRL can obtain state information of the physical networks from edge twin models for training. Meanwhile, the variations of state information can be monitored by edge twin models, which can be utilized to continuously update DRL. Nonetheless, there may exist error between the monitored state information and the real state information (i.e., virtual-real mapping error [35]), due to limitations of digital twin modeling and data collections in IIoT. Proximal policy optimization (PPO) is one of the policy-based DRL approaches. The main characteristic is that an actor module and a critic module iteratively perform training by using policy gradient method, so as to find optimal training parameter. By setting confidence interval, the update range of the training parameter is limited within the confidence interval, which enables PPO well adapt to the state information error and improve stability [36]. However, how to apply PPO approach to solve resource scheduling problem for the integrated federated learning and DAG blockchain with sharding in digital twin-driven IIoT, has not been well studied. Our main contributions in this article are as follows.

1) We develop a new digital twin-driven IIoT framework that integrates federated learning and DAG blockchain with sharding, for enabling secure edge twin model construction. The developed framework includes the data plane, the blockchain plane and the digital twin plane. The functions of these three planes are also illustrated.
2) We propose an efficient resource scheduling scheme for the developed digital twin-driven IIoT while considering both time delay and limited resource capabilities of cluster heads and committees. Accordingly, we formulate an optimization problem to maximize the total utility of the digital twin-driven IIoT.
3) Taking into account the time-varying and stochastic IIoT wireless environment, and the mapping error in

the digital twin plane, we reformulate the optimization problem as a Markov decision process (MDP) and propose multiagent PPO (MAPPO) approach to find the optimal policies of resource scheduling for both federated learning and DAG blockchain with sharding.

The remainder of this article is organized as follows. In Section II, the related works are discussed. In Section III, the proposed digital twin-driven IIoT is introduced. In Section IV, an efficient resource scheduling scheme and utility maximization problem are described. In Section V, an MAPPO approach is provided to solve the optimal solutions. In Section VI, the performance of the proposed approach is provided through extensive simulations. Section VII concludes this article.

II. Related Works

A. Federated Learning and Blockchain for Digital Twin-Driven IIoT

Federated learning is one of the most promising distributed machine learning framework to mitigate the privacy concerns for factory sensitive data in digital twin-driven IIoT. In [15], federated learning was leveraged to construct twin models of physical devices in IoT based on their running data. Moreover, an asynchronous model update scheme was formulated to mitigate the communication overhead. In [16], federated learning was combined with digital twin in air–ground networks to reconcile the conflict between privacy protection and data training. Furthermore, a dynamic incentive scheme was designed to adaptively adjust the selection of the optimal clients and to improve learning accuracy. In [38], digital twin was utilized to assist federated learning by capturing the characteristics of industrial devices. Then, Lyapunov dynamic deficit queue adaptively adjusted the aggregation frequency of federated learning, to improve the learning performance under the resource constraints.

Additionally, blockchain is a distributed and decentralized solution to enable secure interactions among untrusted individuals by maintaining a tamper-proof ledger, which can be utilized to record and verify the model updates of federated learning to strengthen security and privacy protection for edge twin model construction. In [24], in each training round of the developed asynchronous model aggregation scheme, the client devices train the local models by etching the latest global model updates in the blockchain. In [25], a blockchain-assisted hierarchical federated learning for Industry 4.0 was presented, where the digital twin was utilized to accurately capture the characteristics of IIoT devices and assist in federated learning process. In [9], a DAG blockchain was proposed to secure both local model updates and global model updates in digital twin edge networks. Furthermore, an incentive resource cooperation scheme was designed for the APs to contribute resource in the model updates verification. In [31], a blockchain sharding-based scaling scheme was proposed to process massive transactions in 6G networks. Then, a resource optimization scheme was designed to achieve data security and resource efficiency. In [33], blockchain sharding was utilized to enable the simultaneous formation of multiple federated learning models, and a resource allocation problem was formulated to meticulously allocate the necessary bandwidth for the selected devices. In [32], an optimization model for maximizing the throughput of the blockchain with sharding was formulated, to obtain the optimal number of shards. The existing network architectures and technologies have not well analyzed the integration of federated learning and DAG blockchain with sharding in digital twin-driven IIoT. Besides, the challenges associated with applying federated learning and DAG blockchain with sharding for edge twin model construction, such as fast confirmation of model updates transactions, while simultaneously guaranteeing the performance of federated learning in time-varying and stochastic IIoT wireless environment have not been well studied.

B. Resource Scheduling in Digital Twin-Driven IIoT

Recently, with the help of digital twin, DRL has been increasingly utilized to solve complex resource scheduling problems for blockchain and federated learning in IIoT scenarios. By leveraging state information of the physical networks in digital twin, DRL can perform model training to obtain optimal scheduling decisions. In [37], an adaptive blockchain sharding mechanism using deep Q-learning network (DQN) was proposed to automate the selection of parameters of vehicular shards. In [38], a digital twin-assisted asynchronous advantage actor–critic (A3C) was developed for dynamically adjusting the training parameters of federated learning in intelligent driving. In [39], a digital twin-based multiagent deep deterministic policy gradient (DDPG) was exploited to find an optimal solution for edge association in IIoT.

Nevertheless, these works did not consider the virtual-real mapping error in digital twin, which may lead to inaccurate decisions for the resource scheduling. In [35], an improved DRL algorithm with PPO was developed in a digital twin enabled sheet metal assembly context, to adapt to the virtual-real mapping error. The optimal adjust locators of the fixture for individual assemblies can be obtained by the developed algorithm. However, the existing works have not specified any explicit application of PPO for resource scheduling in digital twin-driven IIoT. We provide a general comparison between the studies of previous works and our works in Table I. Motivated by the limitations of previous works, in this article, we first develop a new digital twin-driven IIoT framework that integrates federated learning and DAG blockchain with sharding, for enabling secure edge twin model construction in IIoT. Then, we propose MAPPO-based resource scheduling in the developed framework, with considering virtual-real mapping error.

III. Digital Twin-Driven IIoT Framework

The proposed digital twin-driven IIoT is shown in Fig. 1, illustrating the data plane, the blockchain plane, and the digital twin plane. The data plane performs data processing and executes federated learning for edge twin model construction, the blockchain plane verifies the model updates transactions by utilizing a hierarchical consensus scheme based on DAG blockchain with sharding, and the digital twin plane is

TABLE I
GENERAL COMPARISON

Previous Works	Digital Twin	DAG Blockchain with Sharding	Federated Learning	Utility	DRL
[15]	✓	✗	✓	✗	✗
[16]	✓	✗	✓	✓	✗
[38]	✓	✗	✓	✗	✗
[24]	✓	✗	✓	✗	✗
[9]	✓	✗	✓	✓	✗
[31]	✗	✓	✗	✓	✗
[33]	✗	✓	✓	✗	✗
[32]	✗	✓	✗	✗	✗
[37]	✗	✓	✗	✗	✓
[39]	✓	✗	✗	✗	✓
[35]	✓	✗	✗	✗	✓
Our Works	✓	✓	✓	✓	✓

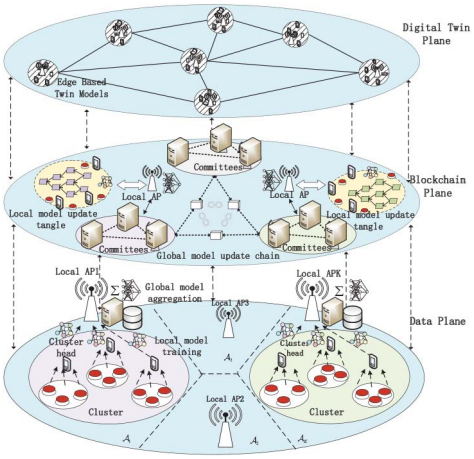


Fig. 1. Architecture of digital twin-driven IIoT.

responsible for constructing and maintaining a virtual replicate of IIoT networks environment, and also designing efficient resource scheduling scheme for both federated learning and DAG blockchain sharding, by observing the state of virtual IIoT environment. The details are as follows.

A. Data Plane

The data plane is shown in the lower part of Fig. 1. We consider there are a set $\mathcal{K}$ of local APs, and each local AP $k \in \mathcal{K}$ is equipped with MEC server, thus has enough computing resource. The local AP $k \in \mathcal{K}$ covers an area $\mathcal{A}_k$, such that $\cup_{\forall k \in \mathcal{K}} \mathcal{A}_k = \mathcal{A}$, and $\mathcal{A}_k \cap \mathcal{A}_{k'} = \emptyset$ for any k, $k' \in \mathcal{K}$ and $k \neq k'$. In each coverage area $\mathcal{A}_k$, the smart devices form a set $\mathcal{U}$ of clusters, and $\mathcal{N}_k^u$ smart devices uniformly distribute in cluster $u \in \mathcal{U}$. The local AP $k \in \mathcal{K}$ has set of $\mathcal{N}_k$ smart devices, and $\mathcal{N} \triangleq \{1, 2, \ldots, N\}$ is the set of all smart devices, where $\mathcal{N} = \cup_{k \in \mathcal{K}} \mathcal{N}_k$. The smart devices generate data packets to characterize various industrial use cases, e.g., robots in assembly process. The data of smart device n is denoted as $\mathcal{D}_n = \{(x_{n1}, y_{n1}), \ldots, (x_{nD_n}, y_{nD_n})\}$, and D_n is the data size. The smart devices periodically synchronize their data with the associated local AP to construct the corresponding edge twin models. In order to reduce energy and bandwidth consumption, the smart device with sufficient resource in each cluster is selected as cluster head for coordinating the data processing. The local AP $k \in \mathcal{K}$ has set of $\mathcal{J}_k$ cluster heads, and $\mathcal{J} \triangleq \{1, 2, \ldots, J\}$ is the set of all cluster heads, where $\mathcal{J} = \cup_{k \in \mathcal{K}} \mathcal{J}_k$.

The edge twin model of smart device n is represented as $DT_n = (\mathcal{M}_n, \mathcal{D}_n, s_n(t), \Delta s_n(t+1))$, where $\mathcal{M}_n$ is the behavior model of the smart device n, $\mathcal{D}_n$ is the static running data, $s_n(t)$ is the real-time dynamic state and $\Delta s_n(t+1)$ is the state update. In order to reduce data leakage risk and create intelligent edge twin model for industrial applications, federated learning is utilized to train model from smart devices' data. In federated learning, each local AP trains task-specific global model individually. The cluster heads in each cluster collect smart devices'data and train local models. Then, the cluster heads synchronously upload the local models to the associated local AP for global aggregation.

B. Blockchain Plane

The blockchain plane is shown in the middle part of Fig. 1, which is responsible for verifying both the local model update and the global model update of federated learning from the data plane. We leverage DAG blockchain with sharding to develop a edge twin model update chain, which consists of two kinds of blockchains: 1) the local model update tangle and 2) the global model update chain. The cluster heads, the smart devices, and the MEC servers act as nodes in the edge twin model update chain. Each node has its unique digital identity consisting of public/privacy keys, to ensure trust in the edge twin model construction. Specifically, the local model update tangle contains all the cluster heads and the smart devices for local model update verification, while the global model update chain contains all of the MEC servers. In order to improve the efficiency for global model update chain, the MEC severs are partitioned into multiple committees (i.e., shards) according to their geographical distribution. Then, the global model update

of the local AP k is recorded with the aggregate local model updates to generate a new sub-block. Each local AP is assigned to access a specific committee for global model update block verification. Therefore, the verification of blocks within the same committee is synchronous and consistent, while the processing among different committees can be asynchronous. For example, the key information of the blocks in each committee, such as block heads, can be established through the transactions among these committees, to compensate for the security degradation raised by the sharding scheme.

Furthermore, a heterogeneous consensus scheme is developed for both the local model update tangle and the global model update chain. Specifically, the cluster heads and the smart devices leverage the DAG with cumulative weight consensus scheme to verify the local model updates, and the MEC servers in each committee utilize the consortium blockchain with delegated Proof-of-Stake (DPoS) consensus scheme to verify the global model update block. The global model update chain and the local model update tangle can interact with each other, in order to form an unified public ledger (i.e., edge twin model update chain). The local APs play the role of cross-chain communicator to transfer the recorded data in these ledgers. For example, in the local model training stage, the cluster heads and the smart devices etch the latest global model updates from the local APs in the global model update chain. In the global model aggregation stage, the local APs retrieve the verified local model updates from the cluster heads and smart devices to execute the global aggregation. The detailed working process for the developed edge twin model update chain will be presented in Section IV.

C. Digital Twin Plane

The digital twin plane is shown in the top part of Fig. 1. The set $\mathcal{K}$ of local APs equipped with MEC servers construct edge twin models of IIoT networks, including the virtual replicates of physical devices and wireless transmission environment. In order to enable the constructed virtual replicate of networks have a function to simulate dynamic changes of physical IIoT networks, the edge twin models maintained by the MEC servers require to synchronize with physical networks constantly to keep consistent with the running states. Besides, the edge twin models in different MEC servers can interact with each other to form digital twin edge networks, providing digital twin services for physical networks with high efficiency. For example, networking schemes by exploiting the appropriate scheduling algorithms can be designed in the digital twin edge networks, to gain the near-real optimization results, and also can be tested before being applied in the physical networks to meet the demands from both the data plane and the blockchain plane. We formulate an efficient resource scheduling scheme for both federated learning and DAG blockchain with sharding, and develop MAPPO approach to solve the problem, which will be illustrated in detail in Section V.

D. Feasibility of Implementation

In real world scenarios, the proposed digital twin-driven IIoT framework that integrates federated learning and DAG blockchain with sharding can be easily implemented in the manufactory industry. The edge twin models of physical devices and IIoT environment are created at MEC servers to obtain accurate running state of the physical situations, and this mapping is synchronized though consistent updates in real-time. The detailed methods for edge twin model construction in real IIoT scenarios have been studied in [40] and [41].

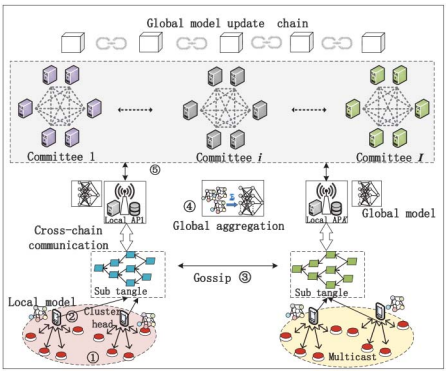


Fig. 2. Working process of the edge twin model update chain.

Besides, the federated learning can be utilized to train task-specific edge twin models for smart automation manufacturing. In the learning process, the smart devices with sufficient resources (i.e., cluster heads) act as clients, to collect data and train local models. Meanwhile, the local APs equipped with MEC servers act as aggregators to aggregate global models. Moreover, a DAG blockchain with sharding is embedded in the learning process, to perform model training both accurately and securely. Specifically, all the cluster heads and the smart devices verify the multicast local model updates by using DAG. Validated local models are aggregated to create an aggregate model at the MEC servers. Then, a block is created for the aggregated models and are added to the blockchain by the committees (sharding of MEC servers). The validated global models are shared with the smart devices, and the cluster heads adjust their local models. The implementation method for federated learning embedded with blockchain has been designed in [42]. Since the constructed edge twin models can replicate the running state of physical situations, they can design resource scheduling scheme, and feedback optimization results to both federated learning and blockchain for execution.

IV. Utility Maximization Problem Formulation

In this section, the work flows and design principles for the edge twin model update chain in digital twin-driven IIoT are first introduced. Then, the problems concerned in resource scheduling are defined. As shown in Fig. 2, the work flows for the developed edge twin model update chain in each epoch are described by the following five steps.

Step 1 (Federated Learning Task Publishing): The AP k distributes a verified global model parameter w_k^{ini} from the global model update chain to its smart devices for training.

Step 2 (Local Model Training): A set $\mathcal{J}_k$ cluster heads act as clients of AP k, and train the local models over the data collected by set of $\mathcal{N}_k$ smart devices.

Step 3 (Local Model Update Verification): The cluster heads and the smart devices in the coverage area of AP k verify the local model updates by leveraging DAG with cumulative weight consensus scheme, to form local model update subtangle. Besides, the subtangles in the different APs needs to be interacted with each other by gossip method to achieve synchronization.

Step 4 (Global Model Aggregation): The AP k receives all the verified local models from the J_k cluster heads, and aggregates the received local models to derive an updated global model w^k.

Step 5 (Global Model Update Verification): The AP k selects one of the committees for adding the updated global model to the global model update chain. The verified global model update is then transmitted back to the J_k cluster heads for the next iteration. Steps 1–5 are repeated until the stopping criteria for federated learning of the local AP $k \in \mathcal{K}$ is reached. In the following, the design principles for the fusion of federated learning and DAG blockchain with sharding are provided.

A. Federated Learning Model

The goal of federated learning is to train a global model $\mathcal{M}_k$ iteratively at the AP $k \in \mathcal{K}$. Here, a synchronous federated learning scheme is considered.

1) Computation Model for Federated Learning: Denote f_{j_k} is the computation resource (i.e., CPU cycle frequency) contributed by the cluster head j_k, and o_{j_k} is the number of CPU cycles required to train one sample of data for the smart devices. Hence, the computation time taken for local iterations of cluster head j_k is denoted as

$$T_{j_k}^{\text{cmp}} = \log\left(\frac{1}{\vartheta_{j_k}}\right)\frac{\sum_{n=1}^{N_k^u} b_n D_n o_{j_k}}{f_{j_k}}, \quad n \in \mathcal{N}_k^u, \quad k \in \mathcal{K} \tag{1}$$

where ϑ_{j_k} is the local accuracy of cluster head j_k, $\log((1/\vartheta_{j_k}))$ is the number of local iterations [43], and b_n is the batchsize for data D_n. Since each AP k is equipped with MEC server and has rich computation resource, the computation time taken for global model aggregation can be neglected.

2) Communication Model for Federated Learning: The cluster head j_k, $j_k \in \mathcal{J}_k$, sends the local model to the AP k by utilizing orthogonal frequency-division multiple access (OFDMA) technology. The cluster head j_k contributes $W_{j_k,k}$ subchannels to send the local model. Therefore, the data rate from cluster head j_k to the AP k can be denoted as

$$R_{j_k,k} = W_{j_k,k} W_0 \log\left(1 + \Gamma_{j_k,k}\right) \tag{2}$$

where W_0 is the bandwidth of each subchannel, $\Gamma_{j_k,k}$ is the signal-to-interference-and-noise ratio (SINR) from the cluster head j_k to the AP k. The size of the trained local model is the same with the shared global model w_k, i.e., $|w_1| = \cdots |w_{j_k}| = |w_k|$. Therefore, the time taken for transmission one updated local model is

$$T_{j_k,k}^{\text{trans}} = \frac{N_k^u |w_k|}{R_{j_k,k}}. \tag{3}$$

Based on the above analysis, the total time taken for one global iteration for the cluster head j_k is

$$T_{j_k}^{gt} = T_{j_k}^{\text{cmp}} + T_{j_k,k}^{\text{trans}}. \tag{4}$$

B. DAG Blockchain With Sharding Model

The DAG blockchain with sharding is developed to strengthen the security for both the local model updates and the global model updates in federated learning. The designed hierarchical consensus model for the edge twin model update chain is as follows.

1) Cumulative Weight-Based Consensus Model: To add a new local model update m_{j_k} to the local model update tangle, the cluster head j_k has to verify two previous unverified transactions by solving a simplified crypto-puzzle. After validation, the cluster head j_k appends the new local model update m_{j_k} to the local model update subtangle by attaching the hashes of two validated transactions to the m_{j_k}. Then, the m_{j_k} enters into cache of the cluster head j_k waiting for multicasting. The m_{j_k} queue in the cache follows first in first out [44]. Moreover, the arrival of the new local model update transactions follows Poisson point process, and λ is the arrival rate. Let H be the number of maximum local model updates for one multicasting. Considering the network traffic load is heavy and the average multicasting time on each cluster head is T^{multi}, the average time taken by the new local model update m_{j_k} for queuing can be denoted by

$$T_{m_{j_k}}^{\text{que}} = \iota T^{\text{multi}} - \frac{H}{2\lambda} \tag{6}$$

where the first item denotes the waiting time for multicasting due to the first in first out queuing, the second item denotes the time counting from the cache having space to it being full, and $\iota \in \mathbb{N}$ denotes cache multiplier. In (6), the average time T^{multi} taken by each cluster head for multicasting can be denoted by

$$T^{\text{multi}} = \frac{1}{J}\left(\sum_{k\in\mathcal{K}}\sum_{j_k\in\mathcal{J}_k} T_{j_k}^{\text{multi}}\right) \tag{7}$$

where $T_{j_k}^{\text{multi}}$ is the multicast time on the cluster head j_k, $j_k \in \mathcal{J}_k$, $T_{j_k}^{\text{multi}} = \max_{j_{k'}\in\mathcal{J}, j_k\neq j_{k'}} (|Hm_{j_k}|/R_{j_k,j_{k'}})$, $|Hm_{j_k}|$ is the size of the disseminated local model update transactions. Considering the cluster head j_k utilizes OFDMA technology

$$\begin{aligned} T_i^{\text{multi}} = {} & \min\left\{\frac{S_i^B}{R_{k,p_i}}, T_{th}\right\} + \min\left\{\max_{v_i\in V_i\setminus\{p_i\}}\left\{\frac{S_i^B}{R_{p_i,v_i}}\right\}, T_{th}\right\} + \min\left\{\max_{v_i,v_i'\in V_i, v_i\neq p_i}\left\{\frac{S_i^B}{R_{v_i,v_i'}}\right\}, T_{th}\right\} \\ & + \min\left\{\max_{v_i,v_i'\in V_i, v_i\neq v_i'}\left\{\frac{S_i^B}{R_{v_i,v_i'}}\right\}, T_{th}\right\} + \min\left\{\max_{v_i,p_i\in V_i}\left\{\frac{S_i^B}{R_{v_i,p_i}}\right\}, T_{th}\right\}. \end{aligned} \tag{5}$$

to disseminate the local model update transactions, R_{j_k,j'_k} is the data rate from the cluster head j_k to the cluster head j'_k, and $R_{j_k,j'_k} = W_{j_k,j'_k} W_0 \log(1+\Gamma_{j_k,j'_k})$, W_{j_k,j'_k} denotes the subchannels allocated by the cluster head j_k, Γ_{j_k,j'_k} is the SINR from the cluster head j_k to the cluster head j'_k.

The cluster head j'_k or the smart device n receives the multicasted local model update transactions and execute verification. If the received local model update transactions are legal, they become new tips and wait for direct or indirect approvement for confirmation. The weight of the local model update transaction m_{j_k} is denoted as $W(m_{j_k})$, which is proportional to the amount of work that the issuing cluster head j_k invested into it, which can be expressed as

$$W_{m_{j_k}} = \frac{\sum_{n=1}^{N_k^u} D_n}{\sum_{n=1}^{N_k} D_n} . \vartheta_{j_k} \tag{8}$$

where $\sum_{n=1}^{N_k^u} D_n$ is the data size accumulated at the cluster head j_k for local training, $\sum_{n=1}^{N_k} D_n$ is the accumulated data size of all smart devices in the coverage of AP k. Moreover, ϑ_{j_k} is the accuracy provided by the issuing cluster head j_k. Furthermore, cumulative weight $CW_{m_{j_k}}$ is used to represent the authenticity of the local model update transaction m_{j_k}, which is the sum of its own weight and its accuracy estimated by other M local model update transactions, which is expressed by

$$CW_{m_{j_k}} = W_{m_{j_k}} + \sum_{j_{k'}=1}^{M} \frac{W_{m_{j_{k'}}}}{\Delta\vartheta_{j_{k'}}} \tag{9}$$

where $\Delta\vartheta_{j'_k} = \vartheta_{j'_k}(m_{j_k}) - \vartheta_{j_k}(m_{j_k})$. $W_{m_{j_{k'}}}$ is the weight of local model update transaction $m_{j'_k}$ that directly or indirectly approves the local model update transaction m_{j_k}.

The confirmation of a local model update transaction depends on its increasing cumulative weight, which is determined by a Markov-Chain Monte Carlo-based tip selection algorithm [45]. Therefore, the time taken for confirmation the local model update transaction m_{j_k} from it is issued by the cluster head j_k to the stage of being confirmed by the local model update tangle can be expressed by

$$T^{\text{conf}}_{m_{j_k}} = T^{\text{que}}_{m_{j_k}} + T^{\text{accu}}_{m_{j_k}} \tag{10}$$

where $T^{\text{que}}_{m_{j_k}}$ is the queuing time for the local model update transaction m_{j_k}, which has been given in (6). $T^{\text{accu}}_{m_{j_k}}$ is the time taken for weight accumulating of the m_{j_k}. By referring to [45], $T^{\text{accu}}_{m_{j_k}}$ can be represented by

$$T^{\text{accu}}_{m_{j_k}} = \left\lfloor 2.84 \cdot \ln\left(2T^{\text{multi}}\lambda\right) \right\rfloor \cdot T^{\text{multi}} + \frac{CW_{th} - CW_{m_{j_k}}}{\lambda} \tag{11}$$

where the first item denotes the time taken for adaption, and the second item denotes the time taken for linear increasing of the $CW_{m_{j_k}}$, and CW_{th} is the confirmation threshold.

2) DPoS-Based Consensus Model: The highly efficient DPoS [46]-based consensus scheme is leveraged in the developed global model update chain. A set $\mathcal{V}$ of delegates in the global model update chain are MEC servers, $\mathcal{V} = \{1, 2, \ldots, V\}$. The delegates are divided into a set $\mathcal{I}$ of committees according to their geographical distribution, $\mathcal{I} = \{1, 2, \ldots, I\}$. Each committee $i \in \mathcal{I}$ has V_i MEC servers that can verify and store the global model update blocks generated by the local APs. In each block verification epoch, a set $\mathcal{K}$ of local APs select one of the committees to verify and store their global model update blocks. Let $\gamma_{i,k}$ denote binary variable such that $\gamma_{i,k} = 1$ when the local AP k selects to access the committee i, and $\gamma_{i,k} = 0$ otherwise. Let $\Omega_i = \{i \in \mathcal{I}, k \in \mathcal{K} | \gamma_{i,k} = 1\}$ denote the local APs that select to access the committee i. After receiving the global model update blocks from the local APs in Ω_i, the committee i generates a new block that contains these global model update blocks, and appends this new block at the end of global model update chain by using DPoS consensus scheme.

In particular, the block manager p_i in committee i, executes global model update block verification and management in its own round of consensus process. Additionally, each delegate v_i, $v_i \in \{\mathcal{V}_i \backslash p_i\}$, is responsible for validating the transactions in the global model update block and replying the audit results to the block manager. The consensus process includes phases of request, pre-prepare, prepare, commit, and reply [23]. The multicast OFDMA is utilized for block transmission in the consensus process. The local AP k, the block manager p_i, and the delegate v_i, $v_i \in \{\mathcal{V}_i \backslash p_i\}$ can be a transmitter in multicast transmission. We consider the block manager p_i allocates W_{k,p_i} subchannels and W_{p_i,v_i} subchannels, the delegate v_i allocates W_{v_i,v'_i} subchannels and W_{v_i,p_i} subchannels. The global model update block size is S_i^B. Hence, the time taken for multicast transmission in the phases of consensus process in the committee i can be expressed in (5), shown at bottom of the previous page, where each summation item denotes the delivery time of the global model update block in each phase of consensus process, respectively, and T_{th} denotes timeout for the delivery of global model update block in each phase. Moreover, R_{k,p_i} is the transmission rate from the local AP k to the block manager p_i, which can be written as $R_{k,p_i} = W_{k,p_i} W_0 \log(1 + \Gamma_{k,p_i})$. R_{p_i,v_i} is the transmission rate from the block manager p_i to the delegate $v_i \in \mathcal{V}_i$, $R_{p_i,v_i} = W_{p_i,v_i} W_0 \log(1 + \Gamma_{p_i,v_i})$. R_{v_i,v'_i} is the transmission rate from the delegate v_i to the other delegate $v'_i \in \{\mathcal{V}_i \backslash v_i\}$, $R_{v_i,v'_i} = W_{p_i,v_i} W_0 \log(1+\Gamma_{v_i,v'_i})$. R_{v_i,p_i} is the transmission rate from the delegate v_i to the block manager p_i, which can be written as $R_{v_i,p_i} = W_{v_i,p_i} W_0 \log(1 + \Gamma_{v_i,p_i})$.

Moreover, in the consensus process of committee i, both the block manager p_i and the delegates v_i need to verify the transactions in the global model update block. The average size of transaction is $\bar{\omega}$. Thus, the number of transactions included in one block is $S_i^B/\bar{\omega}$. We denote the block manager p_i affords o_{p_i} computation load, and allocates f_{p_i} computation resource for verification. The delegate v_i affords o_{v_i} computation load, and allocates f_{v_i} computation resource for verification. The time taken for verifying the global model update transactions and block in the consensus process can be expressed by

$$T_i^{\text{veri}} = \max_{v_i \in V_i} \left\{ \frac{o_{v_i}}{f_{v_i}} \right\}. \tag{12}$$

Therefore, the time taken for the global model update block confirmation in the committee i includes two parts, one is the

multicast transmission time delay T_i^{multi}, and the other one is verification time delay T_i^{veri}, which can be denoted by

$$T_i^{\text{conf}} = T_i^{\text{multi}} + T_i^{\text{veri}}. \tag{13}$$

C. Problem Formulation

It is notable that according to (1)–(3) and (6)–(9), if the cluster heads allocate more computation and spectrum resource for local model training and verification, the time taken for completing one global iteration (4) and the time taken for confirming the local model updates (10) will be reduced. Besides, it can be seen that in (5) and (12), if the block manager p_i and the delegates $v_i \in \mathcal{V}_i$ in the committee i allocate more spectrum resource for global model update block multicasting transmission and contribute more computation resource in block verification, the time spent for global model update block confirmation in (13) will also be reduced. Nevertheless, it causes resource costs for both the cluster heads and delegates which have limited resource. Thus, in order to balance the time delay of federated learning and DAG blockchain with sharding and their resource costs, the utility of the local AP k in the process of local model update verification can be defined as

$$U_{AP_k}^{\text{loc}} = \sum_{j_k=1}^{J_k} \left(B^{\text{loc}} e^{-A^{\text{loc}} T_{j_k}} - \mu_c^{\text{loc}} W_{j_k} W_0 - \varsigma_c^{\text{loc}} f_{j_k} \right) \tag{14}$$

where $B^{\text{loc}} e^{-A^{\text{loc}} T_{j_k}}$ denotes the AP k benefit brought by cluster head j_k in local model update and verification for one global iteration, T_{j_k} is the time taken by cluster head j_k for local model update and verification, $T_{j_k} = T_{j_k}^{gt} + T_{m_{j_k}}^{\text{conf}}$. Besides, B^{loc} represents the initial benefit, and A^{loc} is the decay speed of the benefit. μ_c^{loc} and ς_c^{loc} denote unit spectrum cost and unit computation resource cost of cluster head j_k, respectively. W_{j_k} and f_{j_k} represent the spectrum resource and the computation resource contributed by cluster head j_k, and $W_{j_k} = \sum_{j_{k'}=1, j_k \neq j_{k'}}^{J_k} W_{j_k, j_{k'}} + W_{j_k, k}$, $j_k \in \mathcal{J}_k$.

The utility of the block manager p_i in the process of global model update block verification can be defined as

$$U_{p_i}^{\text{glo}} = \sum_{k=1}^{K} \gamma_{i,k} \begin{pmatrix} B^{\text{glo}} e^{-A^{\text{glo}} T_i^{\text{conf}}} - \mu_c^{\text{glo}} \sum_{v_i=1}^{V_i} W_{v_i} W_0 \\ -\varsigma_c^{\text{glo}} \sum_{v_i=1}^{V_i} f_{v_i} \end{pmatrix} \tag{15}$$

where $B^{\text{glo}} e^{-A^{\text{glo}} T_i^{\text{conf}}}$ denotes the block manager p_i benefit brought by V_i delegates in global model update block verification, T_i^{conf} is the time taken by V_i delegates for global model update block verification. B^{glo} represents the initial benefit, and A^{glo} is the decay speed of the benefit. In addition, μ_c^{glo} and ς_c^{glo} denote unit spectrum cost and unit computation resource cost of delegate v_i, respectively. W_{v_i} and f_{v_i} represent the spectrum resource and the computation resource contributed by delegate v_i, $v_i \in \mathcal{V}_i$.

In our proposed utility optimization problem, the goal of the digital twin-driven IIoT networks is to simultaneously maximize the utilities involved in the local model update verification and in the global model update verification for the edge twin model construction. Therefore, the utility optimization problem can be formulated as

$$\begin{aligned} \max_{\substack{\mathbf{W}^{\text{loc}}, \mathbf{f}^{\text{loc}} \\ \Lambda, \mathbf{W}^{\text{glo}}, \mathbf{f}^{\text{glo}}}} \quad & \sum_{k=1}^{K} U_{AP_k}^{\text{loc}} + \sum_{i=1}^{I} U_{p_i}^{\text{glo}} \\ \text{s.t. C1:} \quad & \sum_{k=1}^{K} \sum_{j_k=1}^{J_k} W_{j_k} \leq W_{\max}^{\text{loc}}, k \in K, j_k \in J_k \\ \text{C2}: \quad & f_{j_k} \leq f_{j_k,\max}^{\text{loc}} \quad \forall j_k \in J_k \\ \text{C3}: \quad & \sum_{i=1}^{I} \gamma_{i,k} = 1 \quad \forall i \in I, k \in K \\ \text{C4}: \quad & \sum_{k=1}^{K} \gamma_{i,k} = \psi_i \quad \forall i \in I, k \in K \\ \text{C5}: \quad & \sum_{i=1}^{I} \sum_{v_i=1}^{V_i} W_{v_i} \leq W_{\max}^{\text{glo}} \quad \forall i \in I, v_i \in V_i \\ \text{C6}: \quad & f_{v_i} \leq f_{v_i,\max}^{\text{glo}} \quad \forall v_i \in V_i \end{aligned} \tag{16}$$

where $\mathbf{W}^{\text{loc}}$ represents bandwidth allocation vector of the cluster head j_k, and $\mathbf{W}^{\text{loc}} = \{W_{1_k}, \ldots, W_{j_k}, \ldots, W_{J_k}\}$, $j_k \in \mathcal{J}_k$. $\mathbf{f}^{\text{loc}}$ represents computation resource allocation vector of the cluster head j_k, and $\mathbf{f}^{\text{loc}} = \{f_{1_k}, \ldots, f_{j_k}, \ldots, f_{J_k}\}$, $j_k \in \mathcal{J}_k$. Λ denotes access indicator vector of the local AP k, $\Lambda = \{\gamma_{i,1}, \ldots, \gamma_{i,k}, \ldots, \gamma_{i,K}\}$, $k \in \mathcal{K}$. $\mathbf{W}^{\text{glo}}$ represents bandwidth allocation vector of the delegate v_i, and $\mathbf{W}^{\text{glo}} = \{W_{1_i}, \ldots, W_{v_i}, \ldots, W_{V_i}\}$, $v_i \in \mathcal{V}_i$. $\mathbf{f}^{\text{glo}}$ represents computation resource allocation vector of the delegate v_i, and $\mathbf{f}^{\text{glo}} = \{f_{1_i}, \ldots, f_{v_i}, \ldots, f_{V_i}\}$, $v_i \in \mathcal{V}_i$. Constraint C1 indicates that all the cluster heads in the coverage of K local APs share the spectrum resource, and the bandwidth allocated by all the cluster heads should not exceed the total bandwidth $W_{\max}^{\text{loc}}$ in the local model update verification process, and C2 ensures that the computation resource contributed by the cluster head j_k should not exceed its maximum computation resource $f_{j_k,\max}^{\text{loc}}$. Constraint C3 restricts that each local AP can access at most one committee, and C4 restricts the maximum number of local APs that can access the committee i is ψ_i. Constraint C5 indicates that all the delegates in the I committees share the spectrum resource, and the bandwidth allocated by all the delegates should not exceed the total bandwidth $W_{\max}^{\text{glo}}$ in the global model update verification process, and C6 ensures that the computation resource contributed by the delegate v_i should not exceed its maximum computation resource $f_{v_i,\max}^{\text{glo}}$.

V. MAPPO Approach for Optimal Resource Scheduling Policies

Considering the dynamic and high-dimensional characteristics of digital twin-driven IIoT networks, it is intractable to solve the problem (16) by exploiting traditional methods. In this section, we propose an MAPPO approach to solve the problem, which is shown in Fig. 3. MAPPO is deployed in the digital twin plane to make sequential resource scheduling decisions for both the local model update verification and the global model update verification. With the help of edge

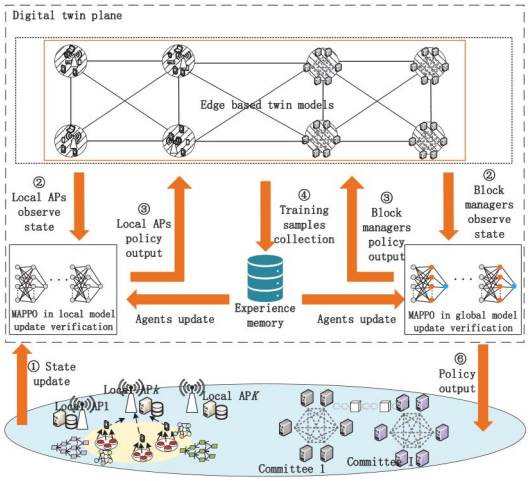


Fig. 3. MAPPO approach for resource scheduling.

twin models, MAPPO can obtain state information of physical objects for training. Although there may exist error between the state information from the edge twin models and the real state information, MAPPO can limit the update range of the training parameters within confidence interval, thus improving the MAPPO's adaptability and stability.

In the proposed approach, the environment of MAPPO consists of edge twin models of cluster heads, smart devices, delegates, and wireless channel. The local AP k, $k \in \mathcal{K}$, in the local model update verification, and the block manager p_i, $i \in \mathcal{I}$, in the global model update verification are considered as multiple agents, respectively. As shown in Fig. 3, in step ①, the physical objects first upload the state updates to their edge twin models for keeping consistency. Then, in step ②, the local AP k, $k \in \mathcal{K}$, and the block manager p_i, $i \in \mathcal{I}$, observe the state of edge twin models, and input the observed state into MAPPO neural networks. Step ③ outputs the policy to solve the utility optimization problem (16), and tests the output policy in the edge twin models. The test value is considered as training samples, and is stored in the experience memory in step ④. The local AP k, $k \in \mathcal{K}$, and the block manager p_i, $i \in \mathcal{I}$ update the MAPPO neural networks by using the training samples in step ⑤, and feedback the optimal policy to the physical objects in ⑥. Comparing with the traditional MAPPO approach without digital twin, in our developed approach, the agents can observe the state of physical objects in the edge twin models, and also test the policy in the edge twin models, which can reduce the resource cost of interaction between agents and physical objects in the MAPPO training process.

A. MAPPO Framework

To implement the proposed approach, we formulate the problem (16) as an MDP, where state space, action space, and reward function are illustrated as follows.

1) State Space: The state space of environment in the local model update verification at decision epoch t consists of the cluster head j_k computation capacity $f^{\text{loc}}_{j_k,\max}$, the SINR $\Gamma_{j_k,k}$ from the cluster head j_k to the local AP k, and the SINR Γ_{j_k,j'_k} from the cluster head j_k to the cluster head j'_k, and $j_k, j'_k \in \mathcal{J}$, $j_k \neq j'_k$. Considering the virtual-real mapping error in the digital twins, the state space of the local AP k agent can be denoted as

$$\begin{aligned}\mathcal{S}^{\text{loc}}_k(t) &= \widetilde{\mathcal{S}}^{\text{loc}}_k(t) + \Delta\mathcal{S}^{\text{loc}}_k(t) \\ &= \left\{\tilde{f}^{\text{loc}}_{j_k,\max} + \Delta f^{\text{loc}}_{j_k,\max}, \tilde{\Gamma}_{j_k,k} + \Delta\Gamma_{j_k,k}, \tilde{\Gamma}_{j_k,j_{k'}} + \Delta\Gamma_{j_k,j_{k'}}\right\}\end{aligned} \tag{17}$$

where $\widetilde{S}^{\text{loc}}_k(t)$ is the accurate state space of the local AP k agent, and $\Delta S^{\text{loc}}_k(t)$ is the state error space. Hence, the state space of K local APs can be denoted as

$$\mathcal{S}^{\text{loc}}(t) = \left\{\mathcal{S}^{\text{loc}}_1(t), \ldots, \mathcal{S}^{\text{loc}}_k(t), \ldots, \mathcal{S}^{\text{loc}}_K(t)\right\}. \tag{18}$$

Meanwhile, the state space of environment in the global model update verification at decision epoch t is composed of the global model update block size S^B_i, the SINR Γ_{k,p_i} from the local AP k to the block manager p_i, the SINR Γ_{p_i,v_i} from the block manager p_i to the delegate v_i, the SINR Γ_{v_i,v'_i} from the delegate v_i to the delegate v'_i, the delegate v_i computation capacity $f^{\text{glo}}_{v_i,\max}$, and the maximum allowable access number of local APs ψ_i in the committee i, where $p_i, v_i, v'_i \in \mathcal{V}_i$, $p_i \neq v_i$, $p_i \neq v'_i$, $v_i \neq v'_i$, $i \in \mathcal{I}$. Similarly, the state space of the block manager p_i can be denoted as

$$\begin{aligned}S^{\text{glo}}_{p_i}(t) &= \tilde{S}^{\text{glo}}_{p_i}(t) + \Delta S^{\text{glo}}_{p_i}(t) \\ &= \Big\{\tilde{S}_{B,k} + \Delta S_{B,k}, \tilde{\Gamma}_{k,p_i} + \Delta\Gamma_{k,p_i}, \tilde{\Gamma}_{p_i,v_i} + \Delta\Gamma_{p_i,v_i} \\ &\quad \tilde{\Gamma}_{v_i,v_{i'}} + \Delta\Gamma_{v_i,v_{i'}}, \tilde{f}^{\text{glo}}_{v_i,\max} + \Delta f^{\text{glo}}_{v_i,\max}, \tilde{\psi}_i + \Delta\psi_i\Big\}\end{aligned} \tag{19}$$

where $\widetilde{S}^{\text{glo}}_{p_i}(t)$ is the accurate state space of the block manager p_i, and $\Delta\mathcal{S}^{\text{glo}}_{p_i}(t)$ is the state error space. The state space of I block managers can be denoted as

$$\mathcal{S}^{\text{glo}}(t) = \left\{\mathcal{S}^{\text{glo}}_1(t), \ldots, \mathcal{S}^{\text{glo}}_{p_i}(t), \ldots, \mathcal{S}^{\text{glo}}_{p_I}(t)\right\}. \tag{20}$$

2) Action Space: In order to maximize utility for digital twin-driven IIoT, the decisions variables need to be adjusted to adapt to the dynamic environment, which includes the bandwidth allocation $\mathbf{W}^{\text{loc}}$ and the computational resource allocation $\mathbf{f}^{\text{loc}}$ in the local model update verification, and also includes the local AP access selection Λ, the bandwidth allocation $\mathbf{W}^{\text{glo}}$ and the computational resource allocation $\mathbf{f}^{\text{glo}}$ in the global model update verification. Therefore, the action space of the local AP k in the local model update verification at decision epoch t can be expressed by

$$\mathcal{A}^{\text{loc}}_k(t) = \left\{\mathbf{W}^{\text{loc}}_k, \mathbf{f}^{\text{loc}}_k\right\}. \tag{21}$$

The action space of K local APs in the local model update and verification at decision epoch t can be expressed by

$$\mathcal{A}^{\text{loc}}(t) = \left\{\mathcal{A}^{\text{loc}}_1(t), \ldots, \mathcal{A}^{\text{loc}}_k(t), \ldots, \mathcal{A}^{\text{loc}}_K(t)\right\}. \tag{22}$$

In the global model update verification at decision epoch t, the action space of the block manager p_i can be expressed by

$$\mathcal{A}^{\text{glo}}_{p_i}(t) = \left\{\mathbf{\Lambda}^{\text{glo}}_{p_i}, \mathbf{W}^{\text{glo}}_{p_i}, \mathbf{f}^{\text{glo}}_{p_i}\right\}. \tag{23}$$

Therefore, the action space of I block managers can be expressed by

$$\mathcal{A}^{\text{glo}}(t) = \left\{\mathcal{A}^{\text{glo}}_1(t), \ldots, \mathcal{A}^{\text{glo}}_{p_i}(t), \ldots, \mathcal{A}^{\text{glo}}_{p_I}(t)\right\}. \tag{24}$$

3) Reward Function: Constraints C1–C6 need to be checked at the end of each iteration in MAPPO. By referring to the penalty function, we define immediate reward $r(t)$ as

$$r(t) = \begin{cases} r^{\text{loc}}(t) + r^{\text{glo}}(t), & \text{when C1} - \text{C6 are satisfied}, \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

where $r^{\text{loc}}(t)$ is the immediate reward of K local APs in the local model update verification, $r^{\text{loc}}(t) = \sum_{k=1}^{K} U_{AP_k}^{\text{loc}}$. Besides, $r^{\text{glo}}(t)$ is the immediate reward of I block managers in the global model update verification, $r^{\text{glo}}(t) = \sum_{i=1}^{I} U_i^{\text{glo}}$. If C1–C6 cannot be met, it indicates the local model update verification and the global model update verification have poor performance. In this case, we set the immediate reward to be zero to avoid the inefficient situation.

In our developed MAPPO framework, both the local AP k agent, $k \in \mathcal{K}$, and the block manager p_i agent, $i \in \mathcal{I}$, adjust their actions to achieve the maximum expected long term reward, which is expressed by

$$\mathcal{R}(t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \ell r\left(\mathcal{S}^{\text{loc}}(t), \mathcal{S}^{\text{glo}}(t), \mathcal{A}^{\text{loc}}(t), \mathcal{A}^{\text{glo}}(t)\right)\right] \tag{26}$$

where ℓ is the discount rate, and $0 \le \ell \le 1$. The training process of the developed MAPPO approach helps both the local AP k agent, $k \in \mathcal{K}$, and the block manager p_i agent, $i \in \mathcal{I}$, find the optimal solutions for problem (16).

B. MAPPO for Resource Scheduling

Since the state and action spaces are high-dimensional data, the developed MAPPO adopts deep neural network combining with reinforcement learning (RL) to address the formulated MDP problem. By constructing the policy optimization model, the MAPPO finds optimal neural network parameter Θ^* to maximize the expected long term reward, which can be denoted as

$$\max_{\boldsymbol{\Theta}} \mathcal{R}(t) = \mathbb{E}_{\xi \sim \pi_{\Theta(\cdot|\mathcal{S})}}\left[\sum_{t=1}^{T} \ell r(t)\right] \tag{27}$$

where ξ is the transition sequence of state and action at decision epoch t, $\pi_{\Theta \sim (\cdot|\mathcal{S})}$ is the policy given state $\mathcal{S}$. The PPO algorithm is one of the RL algorithms which is based on policy gradient and confidence interval [36]. The PPO optimization model consists of Actor and Critic neural networks, where Actor network receives current state and outputs the corresponding decisions, while Critic network estimates the decisions of Actor network and feedbacks estimation results. Actor and Critic networks execute iteratively training by policy gradient to find the optimal parameters $\boldsymbol{\Theta}^*$. The parameters update is limited to the confidence interval in order to improve the PPO's adaptability and stability.

In the training process for local model update verification, each local AP agent obtains initial state $\mathcal{S}^{\text{loc}}(1)$ from edge twin models to generate random policy. Then, each local AP agent interacts with the edge twin models to get updated state and action transition sequence, and adjusts its own actor–critic parameters $\boldsymbol{\Theta}_k^{\text{actor}}$, $\boldsymbol{\Theta}_k^{\text{critic}}$, $k \in \mathcal{K}$. Taking the local AP k agent for example, the loss function in the update process of the Actor network parameter $\boldsymbol{\Theta}_k^{\text{actor}}$ can be denoted as

$$\mathcal{L}\left(\boldsymbol{\Theta}_k^{\text{actor}}\right) = \min\left(\beta\left(\boldsymbol{\Theta}_k^{\text{actor}}\right)\alpha_k(t)\right. \\ \left.\text{clip}\left(\beta\left(\boldsymbol{\Theta}_k^{\text{actor}}\right), 1-\varepsilon, 1+\varepsilon\right)\alpha_k(t)\right) \tag{28}$$

where $\beta(\boldsymbol{\Theta}_k^{\text{actor}})$ is the update range of neural network parameters, $\varepsilon \in (0, 1)$ is a parameter which determines the confidence interval $(1-\varepsilon, 1+\varepsilon)$. clip$(\cdot)$ function limits $\beta(\boldsymbol{\Theta}_k^{\text{actor}})$ to the confidence interval to guarantee the adaptability and stability of Actor network. $\alpha_k(t)$ is advantage function, which represents the advantage value of decisions $\mathcal{A}_k^{\text{loc}}(t)$ compared with other feasible decisions. Specifically, when $\alpha_k(t) > 0$, $\mathcal{A}_k^{\text{loc}}(t)$ can obtain larger expected reward, and $\alpha_k(t) < 0$ otherwise. In (28), $\beta(\boldsymbol{\Theta}_k^{\text{actor}})$ can be denoted as

$$\beta\left(\boldsymbol{\Theta}_k^{\text{actor}}\right) := \frac{\pi_{\boldsymbol{\Theta}_k^{\text{actor}}}\left(\mathcal{A}_k^{\text{loc}}(t)|\mathcal{S}_k^{\text{loc}}(t)\right)}{\pi_{\boldsymbol{\Theta}_k^{\text{actor}},\text{old}}\left(\mathcal{A}_k^{\text{loc}}(t)|\mathcal{S}_k^{\text{loc}}(t)\right)} \tag{29}$$

where $\boldsymbol{\Theta}_k^{\text{actor}}$ represents the current updated Actor network parameters, $\boldsymbol{\Theta}_k^{\text{actor}}$, old represents the old Actor network parameters. Furthermore, in (28), $\alpha_k(t)$ can be denoted as

$$\alpha_k(t) = \delta_k^{\text{critic}}(t) + \ell\delta_k^{\text{critic}}(t+1) + \cdots + \ell^{T-t+1}\delta_k^{\text{critic}}(T-1) \tag{30}$$

where $\delta_k^{\text{critic}}(t)$ represents single-step time error, which can be denoted as

$$\delta_k^{\text{critic}}(t) = r^{\text{loc}}(t+1) + \ell\chi\left(S^{\text{loc}}(t+1)\right) - \chi\left(S^{\text{loc}}(t)\right) \tag{31}$$

where $r^{\text{loc}}(t+1)$ is the immediate reward for decisions $\mathcal{A}_k^{\text{loc}}(t)$, $\chi(\cdot)$ is the estimated reward for decisions $\mathcal{A}_k^{\text{loc}}(t)$. Meanwhile, Critic network takes mean-square error of $\delta_k^{\text{critic}}(t)$ as loss function, and minimizes the loss function by updating the network parameters $\boldsymbol{\Theta}_k^{\text{critic}}$, so that the estimated reward $\chi(\cdot)$ is more accurate. Actor–critic network utilizes adaptive learning rate optimization algorithm for stochastic gradient iterative training, in order to obtain optimal network parameters $\boldsymbol{\Theta}_k^{\text{actor}*}$ and $\boldsymbol{\Theta}_k^{\text{critic}*}$.

Similarly, in the training process of global model update verification, each block manager p_i agent interacts with the edge twin models to obtain state and action transition sequence, and adjusts its own actor–critic parameters $\boldsymbol{\Theta}_{p_i}^{\text{actor}}$ and $\boldsymbol{\Theta}_{p_i}^{\text{critic}}$, $i \in \mathcal{I}$. The block manager p_i agent, $i \in \mathcal{I}$, inputs $\mathcal{S}_{p_i}^{\text{glo}}(t)$ into Actor network to get the corresponding decisions $\mathcal{A}_{p_i}^{\text{glo}}(t)$, and calculates the loss functions $\mathcal{L}(\boldsymbol{\Theta}_{p_i}^{\text{actor}})$ and $\mathcal{L}(\boldsymbol{\Theta}_{p_i}^{\text{critic}})$ in a similar manner to (28) and (31), so as to update network parameters $\boldsymbol{\Theta}_{p_i}^{\text{actor}}$ and $\boldsymbol{\Theta}_{p_i}^{\text{critic}}$, $i \in \mathcal{I}$.

The proposed MAPPO for resource scheduling in local model update verification and in global model update verification are summarized in Algorithms 1 and 2, respectively. Each local AP k agent, $k \in \mathcal{K}$, and each block manager p_i agent, $i \in \mathcal{I}$, initialize their Actor and Critic neural networks. Then, each local AP k agent and each block manager p_i agent choose actions $\mathcal{A}_k^{\text{loc}}(t)$ and $\mathcal{A}_{p_i}^{\text{glo}}(t)$, and execute the actions in the digital twin plane. The obtained immediate reward $r^{\text{loc}}(t)$, $r^{\text{glo}}(t)$, and the next states $\mathcal{S}_k^{\text{loc}}(t+1)$, $\mathcal{S}^{\text{loc}}(t+1)$, $\mathcal{S}_{p_i}^{\text{glo}}(t+1)$, $\mathcal{S}^{\text{glo}}(t+1)$ are calculated, and are stored into

Algorithm 1: MAPPO in Local Model Update Verification

for *Each local AP k agent,* $k \in \mathcal{K}$ **do**
  Initialize the Actor and Critic neural networks parameters $\boldsymbol{\Theta}_k^{actor}$ and $\boldsymbol{\Theta}_k^{critic}$, respectively;
end
for *Each local AP k agent,*$k \in \mathcal{K}$ **do**
  for *Each episode* **do**
    Initialize the digital twin environment;
    for *Each decision epoch t* **do**
      Choose action $\mathcal{A}_k^{loc}(t)$ based on $\boldsymbol{\Theta}_k^{actor}$;
      Execute action in the digital twin plane, obtain immediate reward $r^{loc}(t)$ and the next observation $\mathcal{S}_k^{loc}(t+1)$ and $\mathcal{S}^{loc}(t+1)$;
      Store $\left\{\mathcal{S}_k^{loc}(t), \mathcal{A}_k^{loc}(t), r^{loc}(t), \mathcal{S}^{loc}(t+1)\right\}$ into the experience memory;
      Calculate the advantage value $\alpha_k(t)$ according to (30);
      Update $\boldsymbol{\Theta}_k^{actor}$ and $\boldsymbol{\Theta}_k^{critic}$ by performing stochastic gradient descent;
      Update $\boldsymbol{\Theta}_{k,old}^{actor}$ of every G steps.
    end
  end
end

Algorithm 2: MAPPO in Global Model Update Verification

for *Each block manager* p_i *agent,* $i \in \mathcal{I}$ **do**
  Initialize the Actor and Critic neural networks parameters $\boldsymbol{\Theta}_{p_i}^{actor}$ and $\boldsymbol{\Theta}_{p_i}^{critic}$, respectively;
end
for *Each block manager* p_i *agent,* $i \in \mathcal{I}$ **do**
  for *Each episode* **do**
    Initialize the digital twin environment;
    for *Each decision epoch t* **do**
      Choose action $\mathcal{A}_{p_i}^{glo}(t)$ based on $\boldsymbol{\Theta}_{p_i}^{actor}$;
      Execute action in the digital twin plane, obtain immediate reward $r^{glo}(t)$ and the next observation $\mathcal{S}_{p_i}^{glo}(t+1)$ and $\mathcal{S}^{glo}(t+1)$;
      Store $\left\{\mathcal{S}_{p_i}^{glo}(t), \mathcal{A}_{p_i}^{glo}(t), r^{glo}(t), \mathcal{S}^{glo}(t+1)\right\}$ into the experience memory;
      Calculate the advantage value $\alpha_{p_i}(t)$ in a way similar with (30);
      Update $\boldsymbol{\Theta}_{p_i}^{actor}$ and $\boldsymbol{\Theta}_{p_i}^{critic}$ by performing stochastic gradient descent;
      Update $\boldsymbol{\Theta}_{p_i,old}^{actor}$ of every G steps.
    end
  end
end

experience memory as training samples. The Actor and Critic neural networks are updated.

VI. Numerical Results

A. Numerical Experiment Setting

We build the numerical experiment on TensorFlow-GPU 2.1.0 using Python 3.6, and run on a workstation with hardware configuration of Intel Core i7-11700 processor, 128-GB memory, and NVIDIA RTX A4000. We conduct our experiment on a real-world data set CIFAR10 [47], which consists of 60 000 images in ten classes, including 40 000 training images, 10 000 valid images, and 10 000 test images. The CIFAR10 data set is shuffled and randomly assigned to the smart devices. Therefore, the training data in our experiment is independent and identically distributed (IID). Learning on this image data set simulates the real IIoT scenarios such as defect detection of products in the manufacturing process and image recognition of smart cameras. This data set is also extensively utilized in related works in IIoT [15], [24]. The convolutional neural network (CNN) is utilized as machine learning model for federated learning. The CNN model has 5×5 convolution layers and one dropout layer, followed by two linear layers. We consider a digital twin-driven IIoT networks scenario consisting of 50 APs, and each AP is equipped with MEC server. The APs with MEC severs are partitioned into two committees, and the rest of the APs train global model of federated learning individually. In the coverage area of local AP k, the smart devices form ten spatially disjoint clusters, and $\mathcal{N}_k^u \in [5, 10]$ smart devices uniformly distribute in the cluster. The selected scenario can support the operation of the integrated digital twin, federated learning and blockchain in IIoT, and this scenario was also studied in related works in IIoT [25], [26]. The main simulation parameters are listed in Table II, where the DAG blockchain with sharding parameters are set according to EoS [46] and Tangle [27]. Furthermore, in view of the wide application of normal distribution in the field of measurement error research, the state errors $\Delta S_k^{\text{loc}}(t)$ and $\Delta S_{p_i}^{\text{glo}}(t)$ are set to be random variables that follow normal distribution.

TABLE II
Parameters

Parameter	Value
CPU cycles o_{j_k}	20 cycles/bit
Number of subchannel	100
Bandwidth of subchannel W_0	30MHz
CPU cycles o_{p_i}	0.02 G cycles
CPU cycles o_{v_i}	0.01 G cycles
Learning accuracy ϑ_{j_k}	[0.1, 0.8]
Batch size b_n	3
Confirmation threshold CW_{th}	200
Arrival rate λ	30
Cache length Θ	10
Size of transaction $\lvert m_{j_k}\rvert$ and ϖ	200B
Size of block S_i^k	[1, 10]MB
Number of delegates V_i	21

For the purpose of comparing with our proposed MAPPO approach in both local model update verification and global model update verification, we consider four baseline schemes.

1) Proposed approach without state error, where the states of both local AP agents and block manager agents observed from the digital twin plane are consistent with the real states.
2) Double deep Q-network (DDQN) approach [23] with state error and without state error, where local AP agents and block manager agents make resource scheduling

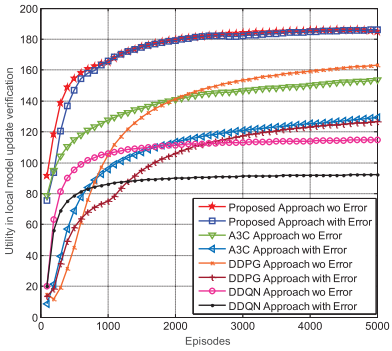


Fig. 4. Error-tolerant performance of different approaches in local model update verification.

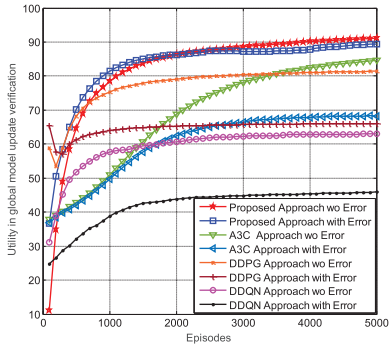


Fig. 5. Error-tolerant performance of different approaches in global model update verification.

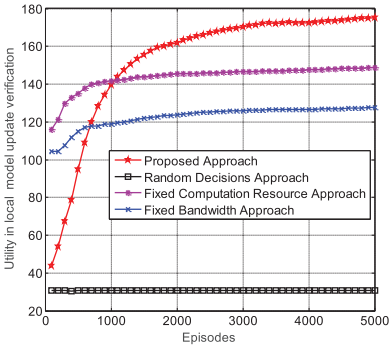


Fig. 6. Utility of the proposed approach with different optimization decisions in local model update verification.

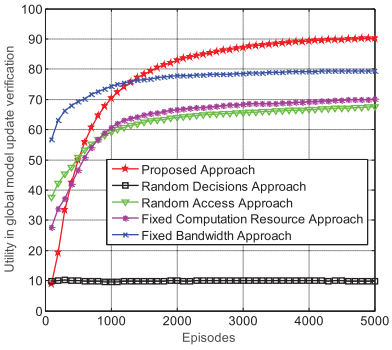


Fig. 7. Utility of the proposed approach with different optimization decisions in global model update verification.

decisions by using DDQN approach, considering the states observed from the digital twin plane with error and without error.

3) DDPG approach [39] with state error and without state error, where local AP agents and block manager agents make resource scheduling decisions by using DDPG approach, considering the states observed from the digital twin plane with error and without error.
4) A3C approach [38] with state error and without state error, where local AP agents and block manager agents make resource scheduling decisions by using A3C approach, considering the states observed from the digital twin plane with error and without error.

B. Performance Analysis

Figs. 4 and 5 show the error tolerant performance of our proposed approach in local model update verification and in global model update verification, respectively. From Fig. 4, we can observe that the utility of the four approaches in local model update verification increases with the training episodes, and finally converges to the fixed points. In the case of the four approaches without error, the utility of proposed approach is the largest, which illustrates that the DDQN, the DDPG and the A3C approaches cannot adapt to the complex environment, that requires the simultaneous execution of both continuous and discrete actions. Moreover, when the four approaches are with error, the utility of the proposed approach is close to that of the case without error. However, the utility of the DDQN, the DDPG and the A3C approaches is lower than that of the case without error, which proves that the proposed approach can well adapt to the error by setting confidence interval. Meanwhile, the utility of DDPG and A3C approaches is almost similar in both the case with error and without error, and the utility of DDQN is the lowest in both the case with error and without error.

In Fig. 5, the utility of the four approaches in global model update verification increases with the training episodes, then converges to the fixed points. Similar to Fig. 4, in the case of the four approaches without error, the utility of proposed approach is the largest, the utility of DDPG and A3C approaches is almost similar, and the utility of DDQN is the lowest. Besides, the utility of the proposed approach with error is close to that of the case without error in global model update verification, and the utility of both the DDQN, the DDPG and the A3C approaches is lower than that of the case without error, which further proves the adaptability and stability of the proposed approach.

Figs. 6 and 7 show the utility of proposed approach with different optimization decisions in local model update verification and in global model update verification, respectively. From Fig. 6, we can see that the utility in local model update verification increases with the training episodes, and converges after 3000 episodes. With the proper computation and bandwidth resource allocation, the utility for local model update verification in (14) can be increased. Hence, the proposed approach receives the highest utility compared with the other three approaches. Besides, we can observe that

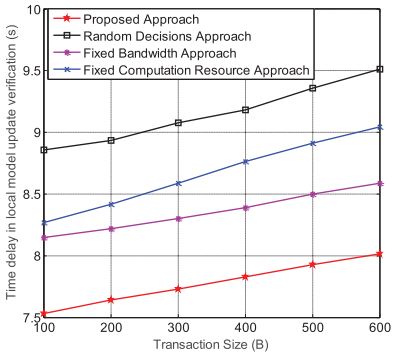


Fig. 8. Time delay of different optimization approaches in local model update verification.

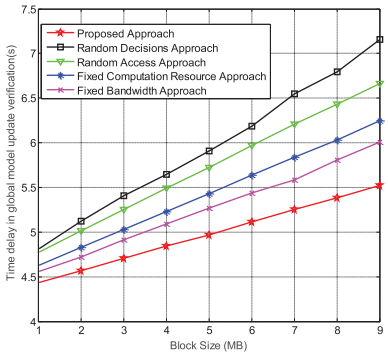


Fig. 9. Time delay of different optimization approaches in global model update verification.

the computation and the bandwidth resource allocation have a significance impact on the time delay for local model update verification in (4)–(11). Hence, the fixed computation and bandwidth resource approaches which cannot optimally schedule the computation and bandwidth resource have the worse utility. Furthermore, in the random decisions approach, the local AP agents randomly schedule the computation and bandwidth resource, which has the worst utility.

From Fig. 7, we can see that the utility in global model update verification is low at the beginning of the training episodes, and it increases and converges after 4000 episodes. Moreover, it can be found that the proposed approach can obtain the highest utility among the four approaches. The reason is that the jointly optimization of local AP access, computation and bandwidth resource allocation for global model update verification can improve the time delay in (13), which contributes to better utility compared with the other four approaches. Additionally, the random access approach, the fixed computation resource approach and the fixed bandwidth approach cannot adaptively schedule the local AP, the computation and the bandwidth resource in the global model update verification process. As a result, the time delay for global model update verification increases, which leads to the worse utility. The random decisions approach has the worst utility.

Fig. 8 shows the time delay of different optimization approaches in local model update verification versus different transaction sizes. It can be observed that the time delay of the four approaches increases with the increasing transaction size. The reason is that the cluster heads can cache less number of transactions for the larger transaction size. Moreover, the multicasting time in (7) and the confirmation time in (10) increase with the increasing transaction size, which leads to the increasing time delay in (14). Besides, our proposed approach can achieve the lowest time delay, then follows the fixed computation and bandwidth resource approach. The random decisions approach has the largest time delay.

Fig. 9 shows the time delay of different optimization approaches in global model update verification versus different block sizes. It can be seen that the time delay increases with the increasing block size, since the larger block size prolongs the delivery time in (5) and the confirmation time in (13) for each sharding. Furthermore, our proposed scheme can achieve the lowest time delay, then follows the fixed bandwidth approach, the fixed computation resource approach, and the random access approach. The random decisions approach has the highest time delay, which proves the advantage of our proposed approach.

VII. Conclusion

This article proposed a new distributed and secure digital twin-driven IIoT framework by integrating federated learning and DAG blockchain with sharding. Specifically, an efficient resource scheduling scheme was designed by considering the performance of both federated learning and DAG blockchain with sharding. Then, an optimization problem was formulated to maximize long-term utility of the digital twin-driven IIoT networks. Furthermore, an MAPPO approach was developed to solve the optimal solutions. Numerical results demonstrated that the developed approach has strong adaptability and stability, and achieves significant improvements in performance of local model update verification and global model update verification. In the future, we will extend our work to adaptively divide the number of shards in the blockchain plane. Besides, we also consider to design distributed federated learning and adaptively select consensus scheme to further improve the performance of the developed digital twin-driven IIoT.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[2] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13789-13804, Sep. 2021.

[3] P. Bellavista, C. Giannelli, M. Mamei, M. Mendula, and M. Picone, "Application-driven network-aware digital twin management in industrial edge environments," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7791–7801, Nov. 2021.

[4] F. Tang, X. Chen, T. Rodrigues, M. Zhao, and N. Kato, "Survey on digital twin edge networks (DITEN) toward 6G," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1360–1381, 2022.

[5] Q. Guo, F. Tang, and N. Kato, "Federated reinforcement learning-based resource allocation for D2D-aided digital twin edge networks in 6G Industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 7228–7236, May 2023.

[6] M. Deng, Z. Yao, X. Li, H. Wang, A. Nallanatghan, and Z. Zhang, "Digital twin assisted UAV dynamic cooperative task assignment based on multi-objective evolutionary algorithm," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3444–3460, Nov. 2023.

[7] H. V. Dang, M. Tatipamula, and H. X. Nguyen, "Cloud-based digital twinning for structural health monitoring using deep learning," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 3820–3830, Jun. 2022.

[8] Y. Lu, S. Maharjan, and Y. Zhang, "Adaptive edge association for wireless digital twin networks in 6G," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16219–16230, Nov. 2021.
[9] L. Jiang, H. Zheng, H. Tian, S. Xie, and Y. Zhang, "Cooperative federated learning and model update verification in blockchain-empowered digital twin edge networks," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11154–11167, Jul. 2022.
[10] W. Sun, H. Zhang, R. Wang, and Y. Zhang, "Reducing offloading latency for digital twin edge networks in 6G," *IEEE Trans. Veh. Tech.*, vol. 69, no. 10, pp. 12240–12251, Oct. 2020.
[11] B. Fan, Y. Wu, Z. He, Y. Chen, T. Q. S. Quek, and C.-Z. Xu, "Digital twin empowered mobile edge computing for intelligent vehicular lane-changing," *IEEE Netw.*, vol. 35, no. 6, pp. 194–201, Nov./Dec. 2021.
[12] M. Groshev, C. Guimarães, J. Martín-Pérez, and A. De La Oliva, "Toward intelligent cyber-physical systems: Digital twin meets artificial intelligence," *IEEE Commun. Mag.*, vol. 59, no. 8, pp. 14–20, Aug. 2021.
[13] B. Fan, Z. Su, Y. Chen, Y. Wu, C. Xu, and T. Q. S. Quek, "Ubiquitous control over heterogeneous vehicles: A digital twin empowered edge AI approach," *IEEE Wireless Commun.*, vol. 30, no. 1, pp. 166–173, Feb. 2023.
[14] J. Lopez, J. E. Rubio, and C. Alcaraz, "Digital twins for intelligent authorization in the B5G-enabled smart grid," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 48–55, Apr. 2021.
[15] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in Industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5709–5718, Aug. 2021.
[16] W. Sun, P. Wang, N. Xu, G. Wang, and Y. Zhang, "Dynamic digital twin and distributed incentives for resource allocation in aerial-assisted Internet of Vehicles," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5839–5852, Apr. 2022.
[17] W. Yang, W. Xiang, Y. Yang, and P. Cheng, "Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1884–1893, Feb. 2023.
[18] Y. Hui et al., "Digital twins enabled on-demand matching for multi-task federated learning in HetVNets," *IEEE Trans. Veh. Tech.*, vol. 72, no. 2, pp. 2352–2364, Feb. 2023.
[19] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technology for the Internet of Things: Research issues and challenges," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2188–2204, Apr. 2019.
[20] M. A. Ferrag, and L. Shu, "The performance evaluation of blockchain based security and privacy systems for the Internet of Things: A tutorial," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17236–17260, Dec. 2021.
[21] M. Shen et al., "Blockchain-assisted secure device authentication for cross-domain Industrial IoT," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 942–954, Mar. 2020.
[22] L. Jiang, S. Xie, S. Maharjan, and Y. Zhang, "Blockchain empowered wireless power transfer for green and secure Internet of Things," *IEEE Netw.*, vol. 33, no. 6, pp. 164–171, Nov./Dec. 2019.
[23] F. Guo, F. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.
[24] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning and permissioned blockchain for digital twin edge networks," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2276–2288, Feb. 2021.
[25] M. Aloqaily, I. A. Ridhawi, and S. Kanhere, "Reinforcing industry 4.0 with digital twins and blockchain-assisted federated learning," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3504–3516, Nov. 2023.
[26] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3690–3700, Aug. 2018.
[27] (IOTA Found., Berlin, Germany) *IOTA: A Cryptocurrency for Internet-of-Things*. (2019). [Online]. Available: http://www.iotatoken.com/
[28] S. Zhu, Z. Cai, H. Hu, Y. Li, and W. Li, "zkCrowd: A hybrid blockchain-based crowdsourcing platform," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4196–4205, Jun. 2020.
[29] L. Jiang, B. Chen, S. Xie, S. Maharjan, and Y. Zhang, "Incentivizing resource cooperation for blockchain empowered wireless power transfer in UAV networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15828–15841, Dec. 2020.
[30] M. Zamani, M. Movahedi, and, M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proc. 2018 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 931–948.
[31] J. Xie, K. Zhang, Y. Lu, and Y. Zhang, "Resource-efficient DAG blockchain with sharding for 6G networks," *IEEE Netw.*, vol. 36, no. 1, pp. 189–196, Jan./Feb. 2022.
[32] Q. Ni, L. Zhang, X. Zhu, and I. Ali, "A novel design method of high throughput blockchain for 6G networks: Performance analysis and optimization model," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25643–25659, Dec. 2022.
[33] H. Moudoud and S. Cherkaoui, "Toward secure and private federated learning for IoT using blockchain," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Rio de Janeiro, Brazil, 2022, pp. 4316–4321.
[34] M. Akbari, M. R. Abedi, R. Joda, M. Pourghasemian, N. Mokari, and M. Erol-Kantarci, "Age of information aware VNF scheduling in Industrial IoT using deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2487–2500, Aug. 2021.
[35] C. Cronrath, A. R. Aderiani, and B. Lennartson, "Enhancing digital twins through reinforcement learning," in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Vancouver, BC, Canada, 2019, pp. 293–298.
[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347v2*.
[37] Y. Lin et al., "DRL-based adaptive sharding for blockchain-based federated learning," *IEEE Trans. Commun.*, vol. 71, no. 10, pp. 5992–6004, Oct. 2023.
[38] L. Tang, M. Wen, Z. Shan, L. Li, Q. Liu, and Q. Chen, "Digital twin-enabled efficient federated learning for collision warning in intelligent driving," *IEEE Trans. Intell. Transp. Syst.*, early access, Dec. 21, 2023, doi: 10.1109/TITS.2023.3330938.
[39] 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 Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5098–5107, Aug. 2021.
[40] V. Kamath, J. Morgan, and M. I. Ali, "Industrial IoT and digital twins for a smart factory: An open source toolkit for application design and benchmarking," in *Proc. IEEE Glob. Internet Things Summit (GIoTS)*, 2020, pp. 1–6.
[41] V. Damjanovic-Behrendt and W. Behrendt, "An open source approach to the design and implementation of digital twins for smart manufacturing," *Int. J. Comput. Integr. Manuf.*, vol. 32, nos. 4–5, pp. 366–384, Mar. 2019.
[42] Y. Li, Y. Lai, C. Chen, and Z. Zheng, " VeryFL: A federated learning framework embedded with blockchain," 2023, *arXiv:2311.15617v1*.
[43] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Paris, France, 2019, pp.1387–1395.
[44] P.-K. Huang, C.-S. Chang, J. Cheng, and D.-S. Lee, "Recursive constructions of parallel FIFO and LIFO queues with switched delay lines," *IEEE Trans. Inf. Theory*, vol. 53, no. 5, pp. 1778–1798, May 2007.
[45] S. Popov, "The Tangle Version 1.4.3," IOTA Foundation, Berlin, Germany, Apr. 2018.
[46] "EOS block producer voting guide." [Online]. Available: https://medium.com/coinmonks/eos-block-producer-voting-guide-fba3a5a6efe0
[47] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," in *Handbook of Systemic Autoimmune Diseases*. Amsterdam, The Netherlands: Elsevier 2009.

Li Jiang (Member, IEEE) received the Ph.D. degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2017.

She was also with the University of Oslo, Oslo, Norway, and Simula Metropolitan Center for Digital Engineering, Oslo, as a visiting Ph.D. student from August 2015 to July 2016. She is currently an Associate Professor with the School of Automation, Guangdong University of Technology, Guangzhou, China. Her current research interests include mobile blockchains, intelligent edge computing, digital twin networks, and resource management for B5G and 6G networks.

Yi Liu received the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2011.

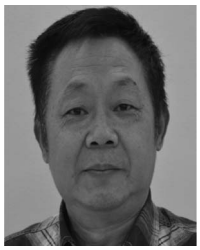
After that, he joined Singapore University of Technology and Design, Singapore, as a Postdoctoral Fellow. In 2014, he worked with the Institute of Intelligent Information Processing, Guangdong University of Technology, Guangzhou, where he is currently a Full Professor. His research interests include wireless communication networks, cooperative communications, smart grid, and intelligent edge computing.

Hui Tian (Senior Member, IEEE) received the M.S. degree in microelectronics and the Ph.D. degree in circuits and systems from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 1992 and 2003, respectively.

She is currently a Professor with BUPT, where she is also the Network Information Processing Research Center Director of the State Key Laboratory of Networking and Switching Technology. Her current research interests include radio resource management, mobile-edge computing, Industrial Internet of Things, and mobile social network.

Lun Tang received the Ph.D. degree in communication and information systems from Chongqing University, Chongqing, China.

He is currently a Professor with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing. His current research interests include next-generation wireless communication networks, heterogeneous cellular networks, and SDN.



Shengli Xie (Fellow, IEEE) received the Ph.D. degree in automatic control from the South China University of Technology (SCUT), Guangzhou, China, in 1997.

He was a Vice Dean of the School of Electronics and Information Engineering, SCUT from 2006 to 2010. He is currently the Director of both the Institute of Intelligent Information Processing and Guangdong Key Laboratory of Information Technology for the Internet of Things, and also a Professor with the School of Automation, Guangdong University of Technology, Guangzhou. He has authored or coauthored four monographs and more than 100 scientific papers published in journals and conference proceedings, and has been granted more than 30 patents. His research interests broadly include statistical signal processing and wireless communications, with an emphasis on blind signal processing and the Internet of Things.