Goal-Conditioned Resource Allocation with Hierarchical Offloading and Reliable Consensus for Blockchain-Based Industrial Digital Twins

Kening Zhang, Carman K. M. Lee, *Senior Member, IEEE*, Yung Po Tsang, *Member, IEEE*

***Abstract*—In the current technological landscape, digital twin (DT) technologies are critical enablers for enhancing communication efficiency and data processing, allowing direct data exchange and on-line monitoring with virtual copies in industry network environments. However, the increasing number of heterogeneous devices and massive sensitive data intensifies challenges in security, management, and service delivery. Rapidly changing business requirements further exacerbate these issues, as traditional algorithms struggle to adapt to dynamic industrial demands. At the same time, overloaded edge servers, ultra-reliable low latency communications (URLLC), and limited resources make real-time decision-making even more difficult. Hence, we propose a hierarchical offloading and resource allocation framework for blockchain-based industrial D2D DT (OR-BIDT). Then, we propose an R-DPoS consensus mechanism that optimizes node selection by introducing a voting mechanism based on transmission reliability and computation frequency to improve the efficiency and security of block verification. For problems requiring optimization over a goal space rather than the simple linear weighted sum in the framework, we design a goal-conditioned reinforcement learning (GCRL) approach with locality sensitive hashing-based experience replay (LSHER) to accomplish efficient experience returns. Simulations show that our proposed algorithm converges faster compared to others, which ensures the accuracy, reliability and security of the OR-BIDT.**

***Index Terms*—Resource allocation, hierarchical offloading, blockchain, digital twin (DT), goal-conditioned reinforcement learning (GCRL), locality sensitive hashing (LSH)**

I. INTRODUCTION

DIGITAL twins (DTs) are an emerging technology that creates a bridge between the physical and digital worlds, which facilitates near-instant and highly reliable wireless communication in Industry 4.0. The advancement of cooperative and predictable computation offloading for real-time, reliable communication in industrial applications depends on the seamless integration of these physical and digital domains [1]. Within this framework, smart devices, particularly in sectors such as smart manufacturing and intelligent logistics, are integral. These devices often generate privacy-sensitive data that can be efficiently managed through industrial Device-to-Device (D2D) communication [2], which allows devices to communicate directly each other, bypassing base stations, conserving transmission power, and reducing the load on network resources [3].

The local processing and integration of resource-intensive data in D2D-based industrial networks present challenges, particularly for lightweight IoT devices that struggle with complex data demands [4], [5]. This leads to inefficiencies and potential disruptions in industrial processes. To mitigate these issues, edge intelligence (EI) has been proposed, offloading selected data to edge and cloud servers for further computation [6]. This approach supports sustainability and aligns with the demands of next-generation industrial networks, such as ultra-reliable low-latency communication (URLLC) and real-time data processing, emphasizing the need for efficient resource allocation (RA).

When physical and digital scenes merge, mapping D2D-based networks to DT space in edge or cloud servers becomes crucial for analyzing and predicting the lifecycle of industrial devices. Although integrating DT and EI enables real-time task offloading, challenges remain, particularly in industrial contexts [7]. First, transmission reliability and data security in industrial DTs are important, but auditing original data and offloading records during transmission is often difficult due to the complex equipment operation [8]. Additionally, intelligent allocation schemes for limited resources are urgently needed to optimize system performance in terms of latency, energy consumption, and throughput [9]. Finally, effective artificial intelligence (AI) algorithms must cooperatively optimize these factors to meet the specific demands of industrial applications, rather than relying on a simple linear summation of weighted objectives.

To address the first challenge, blockchain-enabled networks have been explored to ensure data security during transmission or offloading [8]–[10], the focus remains largely limited to security attributes, with insufficient attention to transmission reliability and the latency requirements of industrial applications. Specifically, existing studies primarily emphasize the immutability and decentralized nature of blockchain to ensure data confidentiality and integrity. However, during resource allocation and task offloading, aspects such as continuity, stability, and latency sensitivity of data transmission are often overlooked. Furthermore, traditional blockchain networks rely on mining processes, which introduce inherent delays

The authors would like to thank the Research and Innovation Office of the Hong Kong Polytechnic University for supporting the project. The work described in this paper is also supported by the project of Research Institute for Advanced Manufacturing, The Hong Kong Polytechnic University *(Corresponding author: Carman K. M. Lee).*

Kening Zhang is with the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: keningcs.zhang@connect.polyu.hk).

Carman K. M. Lee and Yung Po Tsang are with the Department of Industrial and Systems Engineering, Research Institute for Advanced Manufacturing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (e-mail: ckm.lee@polyu.edu.hk; yungpo.tsang@polyu.edu.hk).

incompatible with the low-latency demands of the industrial environment. These drawbacks collectively constrain the potential of blockchain in highly dynamic industrial contexts. Moreover, current algorithms in DT do not simultaneously optimize latency, energy consumption, and throughput, as they often rely on merely assigning weights and linearly combining objectives, which lacks practical significance [11]. To this end, our paper introduces a new hierarchical offloading and resource allocation algorithm for blockchain-integrated industrial D2D-based digital twins (OR-BIDT). The proposed hierarchical offloading mode in DT networks integrates a goal-conditioned reinforcement learning (GCRL) model for decision-making and leverages edge servers and the cloud server (CS) to enhance efficiency. The key contributions of this paper are as follows:

1) We propose an innovative hierarchical data **O**ffloading and **R**A framework for **B**lockchain-enabled D2D **DT** (OR-BIDT). OR-BIDT adopts the D2D network and industrial DT system, in which the hierarchical offloading mode is developed in D2D-edge-cloud network to realize the real-time interaction and optimal resource control between physical and the digital worlds.
2) A **R**andomly **D**elegated **P**roof-**o**f-**S**take consensus mechanism (R-DPoS) is presented in OR-BIDT, which depends on transmission reliability and stakes to select nodes. Then, the success density moderation factor is added to prevent nodes with high computation power and stakes from consistently dominating mining rights.
3) We formulate the RA problem as a goal-augmented Markov decision process (GA-MDP), where total reward can be calculated through the current distance between the system states and multiple goals instead of a simple linear summary of objective functions. A **L**ocality **S**ensitive **H**ashing-based **E**xperience **R**eplay (LSHER) algorithm is designed to prioritize the playback of experiences with the lowest similarity to ensure convergence of this GA-MDP.
4) The experimental results prove the effectiveness of the proposed algorithm and other benchmark policy algorithms in OR-BIDT, which provides the basic framework for new applications of DT technology.

The remainder of this paper proceeds as follows. Section II summarize related works. The OR-BIDT system is proposed in Section III. Then, Section IV formulates the problem. Section V expresses the algorithm. In Section VI, the performance of experiments is illustrated and discussed in detail. Finally, Section VII concludes the paper.

II. RELATED WORKS

A. RA in Industrial DT

Industrial DT divides an entity into a physical world and a virtual world, where various types of data in the physical world can be mapped to the digital space, and then analyses in the digital world can guide the operation of the physical system [12]. DT has been extensively deployed in smart city, smart scheduling, and resource management, which greatly improves the efficiency of real-time decision-making in systems [13]. Several works have been presented in the field of wireless communication. For instance, a relay-based digital twin edge network (DITEN) is presented in [14], which integrated blockchain and resource management to make real-time decisions. Similarly, [7] also designed DITENs and adopted the federated learning technology to optimize specified resources. Moreover, Dong et al. [15] utilized real network data and proposed an offline training DL algorithm in a DT-based system, which reduced the complexity of the algorithm and optimized the normalized variables. Dai et al. [16] proposed an asynchronous actor-critic based DRL algorithm to solve the RA problem with stochastic tasks offloading of IIoT systems in the DT network.

In D2D-based cellular network, multiple technologies are intensively studied for the communication efficiency [17]. Meanwhile, the computation offloading mechanism has been critically discussed in academic circles for MEC and edge-cloud systems. Hong et al. [18] minimized the network latency under a quality-of-service-based computation offloading issue. Cao et al. [19] maximized the energy efficiency with the computation restriction to offload tasks in MEC systems. Guo et al. [20] optimized both energy consumption and latency for data offloading and RA. For the combination of edge computing and AI, EI utilizes intelligent algorithms to address complex edge computing problems [21], and multiple RA strategies based on this framework have been proposed [16], [22]. For example, Lin et al. [21] presented a new approach to allocate resources and address the composite service placement decision issue and in edge networks. Dai et al. [16] designed a novel data offloading and RA approach to optimize the total reward with DRL in end-edge–cloud orchestrated network. Additionally, Zhao et al. [23] studied the total time delay and energy cost in the three-level network, in which the bandwidth, offloading decision, and relay selection are considered.

These methods have combined EI with RA very well in terms of data offloading, but the security and privacy are not considered in the network. Hence, multiple works integrate blockchain technology into MEC systems to ensure the data integrity and validity in the next-generation edge computing enabled with blockchain. Liu et al. [24] studied an adaptive block size in the new blockchain-enabled MEC system. Based on the application level, vehicle blockchain is presented by Kang et al. [25] for data management in edge networks. Xu et al. [26] modelled the blockchain-based MEC system into a crowd-intelligence ecosystem to increase the network efficiency. Asheralieva and Niyato [8] designed a blockchain-enabled RL algorithm to address the RA problem in edge computing, in which the sense of Blockchain as a Service (BaaS) was applied. Xevgenis et al. [27] applied the financial feature of blockchain to implement resource trade on a cloud platform. Wang et al. [28] proposed an adaptive RA approach in the wireless network incorporated with blockchain. Feng et al. [29] implanted a cooperative computation offloading DRL approach for RA in blockchain-based MEC network. Wang et al. [11] applied EI technology in the blockchain-empowered DT network to ensure that the system can allocate resources in real time.

B. Research Gap

While the above studies have made great scholarly contributions in the field of blockchain-based industrial DT, several critical gaps warrant further exploration. There are three primary points where current research falls short, which needs innovative approaches to address these challenges.

Firstly, traditional blockchain mining mechanisms are often ill-suited for the stringent low-latency requirements characteristic of industrial applications. The inherent design of many blockchain protocols prioritizes security and decentralization, which can lead to significant delays in transaction processing. This latency is particularly problematic in industrial settings where real-time data processing and decision-making are crucial. Furthermore, the energy consumption associated with mining activities not only escalates operational costs but also raises sustainability concerns. As industries increasingly seek to optimize their resource allocation, the inefficiencies introduced by conventional mining practices necessitate a reevaluation of blockchain's applicability in these environments.

Secondly, another significant oversight in current research is the lack of consideration for transmission reliability within blockchain frameworks. While blockchain can provide assurances in terms of security, it is incapable of preventing data corruption. In industrial applications, the reliability of data transmission is paramount, as any loss or corruption of data can lead to severe operational disruptions. This gap highlights the need for frameworks that integrate reliable transmission mechanisms into blockchain systems, ensuring that data integrity is maintained throughout the process.

Lastly, previous research has predominantly approached optimization in RA by linearly aggregating smaller objectives, such as minimizing time and energy consumption. However, this method often lacks practical significance, as it fails to capture the complex interdependencies and trade-offs inherent in industrial applications. A more holistic approach to optimization is required, which should consider multiple goals simultaneously and reflect the real-world constraints and requirements of industrial environments.

III. SYSTEM MODEL

In this section, we present the system architecture, and then fully introduce network module, transmission module, computation module and blockchain module, respectively.

A. System Architecture

Fig. 1 presents a blockchain-enabled D2D-based industrial DT architecture where RA is crucial for optimizing productivity and efficiency in distributed systems [17]. The proposed system intelligently assigns servers and offloads tasks based on the current state, adapting to various goals like maximizing throughput or minimizing latency and cost. In the D2D and edge layers, end devices (EDs) offload selected data to edge server nodes (ESNs) for processing. Let $\mathcal{X} = \{1, 2, ..., x\}$ and $\mathcal{Y} = \{1, 2, ..., y\}$ represent EDs and ESNs, respectively. When ESNs reach capacity, tasks are offloaded to the CS. Processed data is either returned to EDs or stored in the CS. In the blockchain system, offloading and storage actions are recorded as transactions, and ESNs participate in R-DPoS consensus to validate transactions as full nodes.

B. Communication Module

For communication from EDs to ESNs, EDs employ orthogonal frequency-division multiple access (OFDMA) to transmit data to edge servers. In this blockchain-supported system, each ED x utilizes η_α subchannels for data transmission, with an uplink bandwidth B_α. The association of subchannel i to ED x is denoted by ϖ_i^x, σ represents noise power, and $g_i^x(t)$ and e_D^T are the channel gain and transmission power of ED x in subchannel i at time slot t. The transmission rate for each ED x is given by:

$$\nu_x^\alpha(t) = B_\alpha \cdot \sum_i^{\eta_\alpha} \varpi_i^x \cdot \log_2 (1 + \frac{g_i^x(t) \cdot e_D^T}{\sigma}) \tag{1}$$

Similarly, when data is transmitted from ESNs to the CS with total η_β subchannels, the transmission rate from the ESN y can be defined as $\nu_y^\beta(t)$ at time slot t, in which B_β is the uplink transmission bandwidth, the ESN y associated to the subchannel i is represented by ω_i^y, and $G_i^y(t)$ and e_E^T mean the channel gain and transmission power of ESN y in subchannel i at time slot t, respectively. $\nu_y^\beta(t)$ can be expressed as:

$$\nu_y^\beta(t) = B_\beta \cdot \sum_i^{\eta_\beta} \varpi_i^y \cdot \log_2 (1 + \frac{G_i^y(t) \cdot e_E^T}{\sigma}) \tag{2}$$

C. Computation Module

In this system, EDs can execute tasks locally or offload them to ESNs. Complex tasks and block mining are sometimes not parallel due to ESN limitations. Therefore, ESNs can further offload tasks to the CS for optimized processing. These offloaded tasks generate computational receipts, such as data hashes and signatures, which are recorded on the blockchain as transactions.

Let $C_{D,x}$ be the CPU cycles required to process tasks on ED x at time slot t, with $f_{D,x}$ as the CPU frequency and $e_{D,x}$ as the computing power. If the ED processes tasks locally, the processing latency at time t, $t_{D,x}^\alpha(t)$, is given by:

$$t_{D,x}^\alpha(t) = \frac{C_{D,x}(t)}{f_{D,x}} \tag{3}$$

the resulting energy $w_{D,x}^\alpha(t)$ generated by the ED x in the above process can be expressed as follows:

$$w_{D,x}^\alpha(t) = e_{D,x} \cdot t_{D,x}^\alpha(t) = \frac{e_{D,x}}{f_{D,x}} \cdot C_{D,x}(t) \tag{4}$$

When an ED offloads computation to an ESN, latency and energy overhead primarily result from data transmission and processing at the ESN. Let e_T represent the transmission power between the ED and ESNs over wireless links. Data is segmented into units of size s_0 for efficient processing, with each segment marked by a unique number to ensure proper ordering. At time slot t, the number of segments from ED x is $N_x^\alpha(t)$, and the CPU frequency and power of ESN y

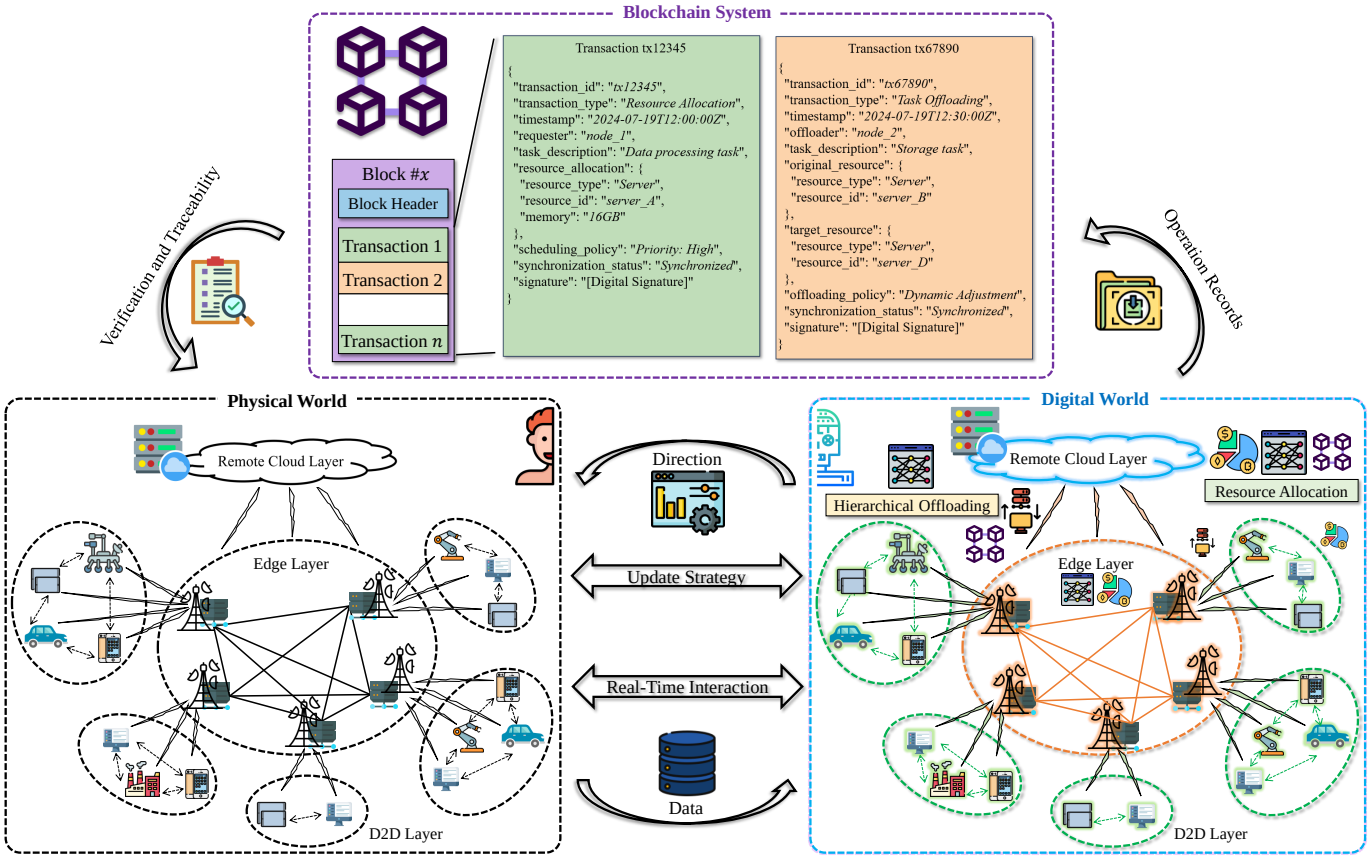


Fig. 1. The Framework of Proposed OR-BIDT System.

are $f_{E,y}(t)$ and $e_{E,y}$, respectively. The latency and energy overhead at time slot t are calculated as follows:

$$t^{\alpha}_{E,x,y}(t) = \frac{s_0}{\nu^{\alpha}_x(t)} \cdot N^{\alpha}_x(t) \ + \frac{C_{D,x}(t)}{f_{E,y}(t)} \tag{5}$$

$$w^{\alpha}_{E,x,y}(t) = \frac{e^T_D \cdot s_0}{\nu^{\alpha}_x(t)} \cdot N^{\alpha}_x(t) + \frac{e_{E,y}}{f_{E,y}(t)} \cdot C_{D,x}(t) \tag{6}$$

where the transmission rate that we use in equations is the average of all EDs for simplified calculations.

In the same way, CPU cycles that are needed to compute tasks in the ESN y at time slot t is presented as $C_{E,y}(t)$. When current computation tasks in ESNs are processed locally, the processing latency can be defined as:

$$t^{\beta}_{E,y}(t) = \frac{C_{E,y}(t)}{f_{E,y}(t)} \tag{7}$$

and energy used for processing tasks can be denoted as:

$$w^{\beta}_{E,y}(t) = e_{E,y} \cdot t^{\beta}_{E,y}(t) = \frac{e_{E,y}}{f_{E,y}(t)} \cdot C_{E,y}(t) \tag{8}$$

If some current computation tasks at the ESN y are offloaded to the CS, the latency and consumed energy can be deduced using the same as the above-mentioned principle, where the transmission rate means the average of all ESNs. Suppose that there are $N^{\beta}_y(t)$ segments transmitted from the ESN y to the CS at time slot t, while f_S and e_S represent the CPU computation frequency and power of the CS, respectively. Consequently, the corresponding results can be calculated by:

$$t^{\beta}_{S,y}(t) = \frac{s_0}{\nu^{\beta}_y(t)} \cdot N^{\beta}_y(t) \ + \frac{C_{E,y}(t)}{f_S} \tag{9}$$

$$w^{\beta}_{S,y}(t) = \frac{e^T_E \cdot s_0}{\nu^{\beta}_y(t)} \cdot N^{\beta}_y(t) + \frac{e_S}{f_S} \cdot C_{E,y}(t) \tag{10}$$

Through probability theory, we define events A and B as "Data of the ED is offloaded" and "Data of the ESN is offloaded," respectively. These events are represented by indicator functions $I_{A,x}$ and $I_{B,y}$, which follow a Bernoulli distribution. $I_{A,x}$ is 1 if event A occurs in x, and 0 otherwise; similarly, $I_{B,y}$ is 1 if event B occurs in y, and 0 otherwise. Thus, the total service time of the system is given by:

$$\begin{aligned}\mathcal{T}^s(t) = \sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} ([t^{\alpha}_{D,x}(t)]^{1-I_{A,x}} \cdot [t^{\alpha}_{E,x,y}(t)]^{I_{A,x}}) + \\ \sum_{y\in\mathcal{Y}} ([t^{\beta}_{E,y}(t)]^{1-I_{B,y}} \cdot [t^{\beta}_{S,y}(t)]^{I_{B,y}})\end{aligned} \tag{11}$$

Note that the time calculated here is the CPU time not the wall-clock time, and the aggregate energy overhead can be represented as:

$$\begin{aligned}\mathcal{W}(t) = \sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} ([w^{\alpha}_{D,x}(t)]^{1-I_{A,x}} \cdot [w^{\alpha}_{E,x,y}(t)]^{I_{A,x}}) + \\ \sum_{y\in\mathcal{Y}} ([w^{\beta}_{E,y}(t)]^{1-I_{B,y}} \cdot [w^{\beta}_{S,y}(t)]^{I_{B,y}})\end{aligned} \tag{12}$$

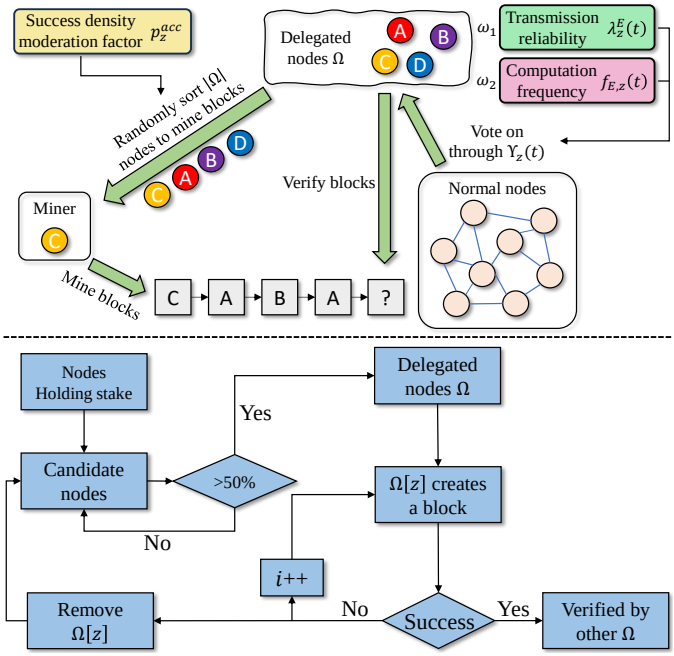


Fig. 2. Process of R-DPoS consensus algorithm.

D. Reliable Transmission

In the proposed system, we give the calculation of the reliability $\lambda_e(t)$ similar to [11], which consists of three components: 1) transmission reliability between EDs and the ESN, 2) transmission reliability between ESNs and the cloud server and 3) transmission reliability between delegated nodes and edge servers, denoted as $\lambda_x^{DE}(t)$ and $\lambda_y^{ES}(t)$ and $\lambda_y^E(t)$, respectively. We utilize φ_x^D, φ_y^E and φ_y^S to represent the transmission link coefficient from EDs to the ESN and transmission link coefficient from ESNs to the cloud server. Then, the transmission reliability can be calculated by $\lambda_x^{DE}(t) = e^{-\varphi_x^D(s_0 \cdot N^\alpha(t)/\nu_x^\alpha(t))}$, $\lambda_y^{ES}(t) = e^{-\varphi_y^E(s_0 \cdot N^\beta(t)/\ \nu_y^\beta(t))}$ and $\lambda_y^E(t) = e^{-(1-p_y^{acc})s_d \cdot N^B(t)/\nu_p(t) - \varphi_y^S \cdot t_p(t)}$. The reliability of the entire system is deduced as:

$$\lambda_e(t) = \prod_{x \in \mathcal{X}} \lambda_x^{DE}(t) \prod_{y \in \mathcal{Y}} \lambda_y^{ES}(t) \prod_{y \in \mathcal{Y}} \lambda_y^E(t) \tag{13}$$

E. Blockchain Module

In the proposed model, ESNs act as blockchain nodes for auditing and storing records of offloading data, and high-reputation nodes are selected through voting similar to [30]. The blockchain module collects and records transactions on the chain, with the set of delegated nodes participating in the blockchain network denoted as $\Omega = \{1, 2, 3, \cdots, z\}$, where $\Omega \subset \mathcal{Y}$. Transactions are securely recorded in two steps: generating blocks and broadcasting them for consensus.

1) Block Generation: A scheduling strategy selects a block producer from full nodes, which then collects, validates, and packages transactions into a new block. Let ρ be the processing density (in CPU cycles/bit), s_d the size of a unit transaction (in bits/unit), and $N^B(t)$ the number of transactions that represents offloading processes, hash-encrypted signatures, and data records generated by EDs and ESNs at time slot t [31]. The time cost by the producer z for block generation can be given by:

$$t_{g,z}(t) = \frac{\rho \cdot s_d}{f_{E,z}(t)} \cdot N^B(t) \tag{14}$$

2) Consensus Process: We present R-DPoS, an enhanced version of DPoS [32], to achieve consensus without the resource-heavy competition of Proof-of-Work (PoW) systems like Bitcoin [33]. As shown in Fig. 2, nodes holding tokens become candidate nodes initially. If a candidate node receives over 50% of the votes, it is promoted to a delegated node. These delegated nodes are then arranged in a randomized sequence to create blocks, which are subsequently submitted for validation by regular nodes. After each block creation, the index of the delegated node increments; if a block fails verification, the responsible delegated node is removed. Additionally, factors such as transmission reliability and computational frequency influence the voting mechanism, determining the nodes eligible to participate in block creation. Finally, the created blocks are confirmed only after successful verification by other delegated nodes.

For easier calculation on the mathematical model, we divide R-DPoS into two parts: block propagation and block verification, in which we define transmission reliability between delegated nodes and edge servers as $\lambda_z^E(t)$. The success density moderation factor of ESNs and transmission link coefficient among edge servers are denoted as p_z^{acc} and φ_z^S, so that $\lambda_z^E(t)$ can be calculated by $\lambda_z^E(t) = e^{-(1-p_z^{acc})s_d \cdot N^B(t)/\nu_p(t) - \varphi_z^S \cdot t_p(t)}$. In order to follow the R-DPoS in the proposed blockchain-enabled system, consensus nodes are chosen based on their high number of votes invested in block generation and verification. The number of votes assigned to delegated nodes for the consensus is determined by holding stakes $\boldsymbol{\Upsilon}^s(t) = \{\Upsilon_1(t), \Upsilon_2(t), \cdots, \Upsilon_z(t)\}$, where $\Upsilon_z(t)$ is determined by $\Upsilon_z(t) = \omega_1 \cdot \lambda_z^E(t) + \omega_2 \cdot f_{E,z}(t)$, where ω_1 and ω_2 are weight factors, $\omega_1, \omega_2 \in (0, 1)$.

In the first process, let I^B and t_l be the block interval and the broadcast latency between ESNs [30]. The rate of wire link from the block producer z to the block producer z^* at time slot t is indicated as $\nu_p(t)$, where $\nu_p(t) = \min_{z^* \in \Omega, z^* \neq z} \left\{ \nu_{z^*}^\beta(t) \right\}$ and $\Omega \subseteq \mathcal{Y}$. Then, the broadcast time of the block in P2P network can be calculated as follows:

$$t_p(t) = \frac{s_d \cdot N^B(t)}{\nu_p(t)} + I^B + t_l \tag{15}$$

In the verification process, the computational cost of cryptographic operations is crucial, as demonstrated in [30]. Let $\theta(t)$ denote the CPU cycles required for block verification at time slot t. The validation time is then calculated by:

$$t_{v,z}(t) = \frac{\theta(t)}{f_{E,z}(t)} \tag{16}$$

The blockchain system's delay is measured by the 'delay/time to finality' (DTF/TTF), which indicates the time needed to finalize transactions. Let C be the event where an ESN is selected as a block producer, and $I_{C,z}$ be an indicator

function that is 1 if C occurs in z, and 0 otherwise. The DTF for transactions at time slot t from z is calculated as:

$$\mathcal{T}^B_{sum}(t) = \sum_{z\in\Omega} (t_{g,z}(t) + t_p(t) + t_{v,z}(t)) \cdot F_{C,z} \tag{17}$$

IV. PROBLEM FORMULATION

In this section, we introduce the problem formulation by leveraging the Markovian property, which is well-known for modeling wireless channels [34], [35]. For the proposed OR-BIDT that involves more general and complex scenarios, we adopt the GA-MDP framework to define the problem and address it with GCRL [36].

A. State Variable and Space

At each discrete time epoch t $(t = 1, 2, \cdots,)$, the state of the system environment is observed first by the agent, the experience is obtained, and then the decision policy is updated through the proposed algorithm. For the system state space, we define five components to describe the current state, which are $\mathcal{U}(t), \mathcal{T}^B(t), \mathcal{Q}(t), \mathbf{\Upsilon}^s(t)$ and $\lambda^s(t)$, respectively. The union of the operational capability in the system $\mathcal{U}(t) = \left\{\mathcal{T}^s(t), \mathcal{W}(t), \mathcal{T}^B_{sum}(t)\right\}$. The current time cost spent by blockchain producers can be denoted as $\mathcal{T}^B(t) = \left\{\mathcal{T}^B_1(t), \mathcal{T}^B_2(t), \cdots, \mathcal{T}^B_z(t)\right\}$.The channel gain from EDs to ESNs and from ESNs to the CS can be denoted as $\mathcal{Q}(t) = \{\mathbf{g}(t), \mathbf{G}(t)\}$, where $\mathbf{g}(t) = \left[\vec{g^1}(t)\ \vec{g^2}(t)\ \cdots\ \vec{g^x}(t)\right]$ and $\mathbf{G}(t) = \left[\vec{G^1}(t)\ \vec{G^2}(t)\ \cdots\ \vec{G^y}(t)\right]$. In $\mathbf{g}(t)$ and $\mathbf{G}(t)$, subchannel gains distributed in each unit can be given by $\vec{g^x}(t) = (g^x_1(t),\ g^x_2(t),\ \cdots, g^x_{\eta_\alpha}(t))$ and $\vec{G^y}(t) = (G^y_1(t), G^y_2(t), \cdots, G^y_{\eta_\beta}(t))$. The union of the current reliability $\lambda^s(t) = \left\{\lambda^{DE}(t), \lambda^{ES}(t), \lambda^E(t), \lambda_e(t)\right\}$, which is composed by three joint vectors $\lambda^{DE}(t) = \left\{\lambda^{DE}_1(t), \lambda^{DE}_2(t), \cdots, \lambda^{DE}_x(t)\right\}$, $\lambda^{ES}(t) = \left\{\lambda^{ES}_1(t), \lambda^{ES}_2(t), \cdots, \lambda^{ES}_y(t)\right\}$, $\lambda^E(t) = \left\{\lambda^E_1(t), \lambda^E_2(t), \cdots, \lambda^E_y(t)\right\}$ and a scalar $\lambda_e(t)$. The number of stakes $\mathbf{\Upsilon}^s(t) = \{\Upsilon_1(t), \Upsilon_2(t), \cdots, \Upsilon_y(t)\}$. Then, the state space is represented as:

$$s(t) \triangleq \left\{\mathcal{U}(t), \mathcal{T}^B(t), \mathcal{Q}(t), \mathbf{\Upsilon}^s(t), \lambda^s(t)\right\} \in \mathcal{S} \tag{18}$$

In our model, the continuous state space renders the likelihood of a specific state meaningless. Let $\mathbf{f}^d(\cdot)$ denote the probability density function. The probability of transitioning from state $s(t)$ to $s(t+1)$ after action $\mathcal{A}(t)$ is taken can be expressed in integral form as follows:

$$\mathcal{P}(s(t+1)|s(t), a(t)) = \int \mathbf{f}^d(s(t), a(t), s_\varepsilon)\ ds_\varepsilon \tag{19}$$

B. Action Variable and Space

We define the action space as $a(t) \triangleq \left\{\mathcal{O}^D(t), \mathcal{O}^E(t), f^E(t)\right\} \in \mathcal{A}$, which consists of hierarchical offloading decisions $\mathcal{O}^D(t)$ and $\mathcal{O}^E(t)$, and CPU frequency allocation decisions for computation $f^E(t)$.

Hierarchical Offloading Decisions: EDs and ESNs can both choose whether to offload tasks for RA. Decisions of EDs and ESNs can be denoted by:

$$\mathcal{O}^D(t) \triangleq \left\{o^D_1(t), o^D_2(t), \cdots, o^D_x(t)\right\} \tag{20}$$

$$\mathcal{O}^E(t) \triangleq \left\{o^E_1(t), o^E_1(t), \cdots, o^E_y(t)\right\} \tag{21}$$

Frequency Allocation Decisions: With multiple tasks to process in ESNs, including those offloaded from EDs, tasks generated within ESNs, and blockchain mining, ESNs must allocate CPU frequency across these tasks to maximize the total reward, as follows:

$$f^E(t) \triangleq \{f_{E,1}(t), f_{E,2}(t), \cdots, f_{E,y}(t)\} \tag{22}$$

where $\sum_{y\in\mathcal{Y}} f_{E,y}(t) = f_{E,max}$, and $f_{E,max}$ is the total CPU frequency in the system.

C. Goal Space

For the GA-MDP, ψ is a mapping function, which maps the state space to the goal space. Here, we adopt the first two elements of the state space $\mathcal{U}(t) = \mathcal{T}^s(t), \mathcal{W}(t), \mathcal{T}^B(t)$ as the goal space, which means $g \in \mathcal{G} = \psi(\mathcal{S}) = \mathcal{U}$.

D. Reward Function

In this paper, the reward goal is to achieve long-term goals $\mathcal{G}$, including total service latency, aggregate energy overhead, block interval, and transaction DTF. Unlike the linear accumulation of metrics in other studies, our reward is defined by states, actions, and goals as follows:

$$\gamma^G(s_t, a_t, g) = \begin{cases} 0, & \|\psi(s_{t+1}) - g\|_2 \le \varepsilon_g \\ -1, & otherwise \end{cases} \tag{23}$$

where s_t and a_t mean $s(t)$ and $a(t)$, ε_g is an any small real number $(\varepsilon_g > 0)$. Given the consideration of the long-run comprehensive reward of the entire system, an optimal policy $\pi(a|s, g)$ in GCRL is needed to find. We denote the desired goal distribution of the environment and the discount factor as $\mathcal{P}^g$ and γ, where $\gamma \in (0, 1)$. Then the goal-conditioned value function in T-step future can be given by:

$$\mathcal{V}^\pi(s_t, g) = \mathbb{E}_{\substack{a_t\sim\pi(\cdot|s_t,a_t) \\ s_{t+1}\sim\mathcal{P}(\cdot|s_t,a_t)}} \left[\sum_t^T \gamma^t r^G(s_t, a_t, g)\right] \tag{24}$$

V. GOAL-CONDITIONED REINFORCEMENT LEARNING ALGORITHM WITH LSHER-BASED GCRL FRAMEWORK

In this section, due to the continuous actions of the system and the rather sparse reward, the LSHER-based GCRL algorithm is developed as shown in Fig. 3. In this framework, an agent interacts with the environment and receives observations, which are processed by the OR-BIDT. The LSH-based state selection module efficiently categorizes states into hash buckets, allowing the algorithm to quickly retrieve and evaluate similar states, which facilitates the selection of optimal actions. The overall system optimizes by iterating through actions, rewards, and updates across both the actor-critic networks and the hash-based state selection, ultimately refining policy and value estimations for improved performance.

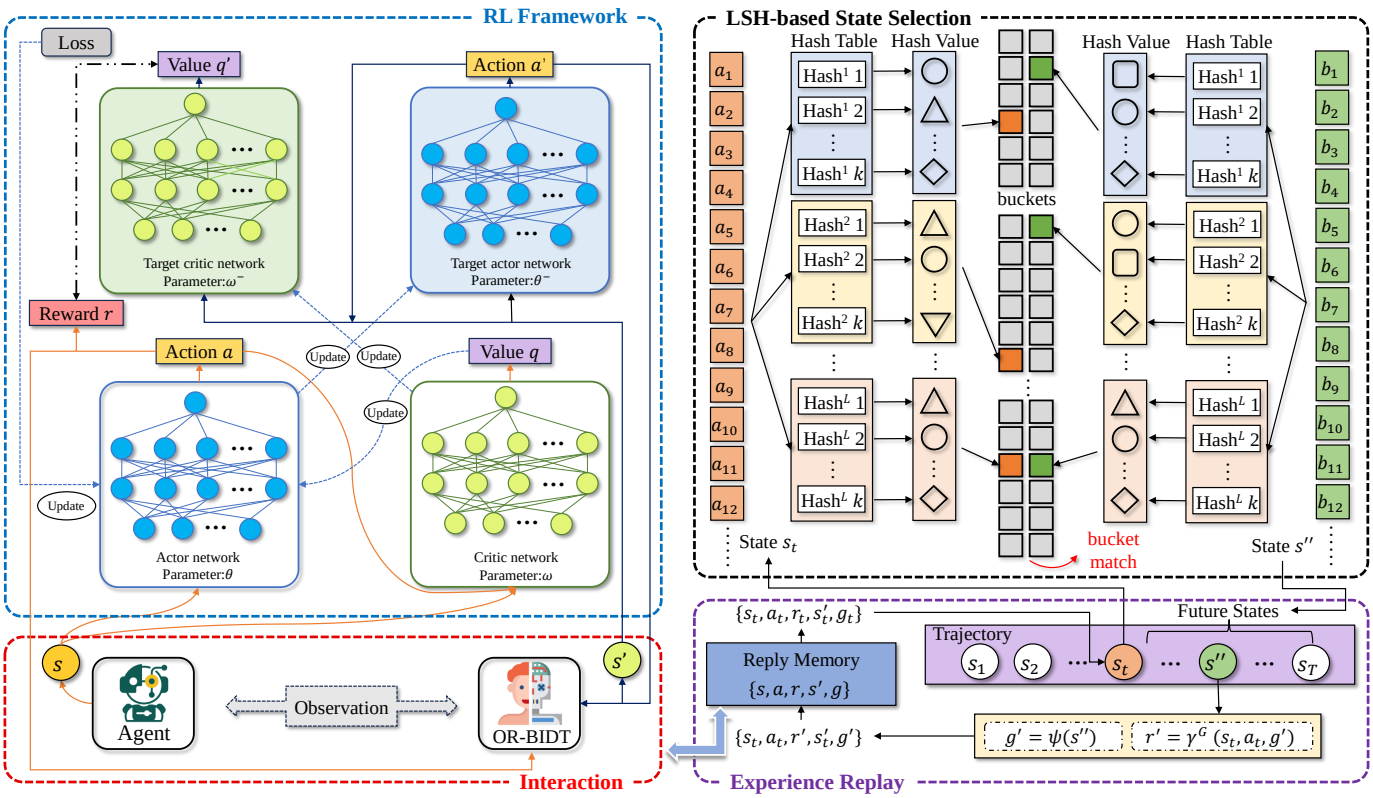


Fig. 3. Process of the proposed algorithm.

A. DDPG Framework for RA Problem in OR-BIDT

Since the actions are defined in continuous space, the DDPG algorithm is used to solve the proposed GCRL problem. This algorithm utilizes the actor-critic structure, incorporating both an actor network and a critic network. The actor network is used to generate a deterministic action after observing the current state, and the critic network is used to evaluate the value of the action based on this state. Due to the estimation bias caused by bootstrapping in TD-error, two other target networks are designed to obtain the action and value in next time epoch during the process of optimising the network parameters, which enhances convergence and stability. After the actor and critic networks are updated, the two target networks are iterated through updated parameters and themselves as well.

Given a goal g during a trajectory, the agent observes the current state s_t of the system at the epoch t, which is inputted into the actor network $\pi(s_t, g; \theta)$, and then a deterministic action a_t is outputted by $a_t = \pi(s_t, g; \theta)$. Note that actions are continuous, so the output of $\pi(s_t, g; \theta)$ here is a deterministic value, not the probability density. The agent calculates the action value q_t by $q_t = q(s_t, a_t, g; \omega)$ at the current epoch t. After the agent performs the action a_t, it receives a reward r_t^G and the system enters the next state s_{t+1}. To prevent estimation errors, two target networks, $\pi(; \theta^-)$ and $q(; \omega^-)$ are used to get the next action a_{t+1} and value q_{t+1}, where $a_{t+1} = \pi(s_{t+1}, g; \theta^-)$ and $q_{t+1} = \Pi(s_{t+1}, a_{t+1}, g; \omega^-)$. Here, a_{t+1} and q_{t+1} are only computed, and the agent do not take the action a_{t+1}. θ, ω, θ^- and ω^- are parameters of four networks.

The quintuple $(s_t, a_t, s_{t+1}, r_t^G, g)$ is a transition, which is recorded in an experience replay pool with capacity N_c for future training. Take a transition $(s_t, a_t, s_{t+1}, r_t^G, g)$ from the buffer, and the more precise value can be obtained from the Bellman equation as follows:

$$y_t = r_{t,g}^G + \gamma q(s_{t+1}, \pi(s_{t+1}, g; \theta^-), g; \omega^-) \tag{25}$$

To train the parameters θ of the actor network, the action value function can be maximized through accumulation in a trajectory as an objective function, i.e.,

$$\mathcal{J}(\theta) = \frac{1}{N_c} \sum_t q(s_t, \pi(s_t, g; \theta), g; \omega) \tag{26}$$

The policy gradient of the actor network can be found by the chain derivative rule:

$$\begin{aligned} \nabla_{\theta} \mathcal{J}(\theta) &= \frac{1}{N_c} \sum_t \frac{\partial q(s_t, \pi(s_t, g; \theta), g; \omega)}{\partial \theta} \\ &= \frac{1}{N_c} \sum_t \frac{\partial q(s_t, a_t, g; \omega)}{\partial a_t} \frac{\partial \pi(s_t, g; \theta)}{\partial \theta} \end{aligned} \tag{27}$$

Denote the learning rate in the actor network as ξ_a, where $\xi_a \in (0, 1)$, and the parameters can be updated by:

$$\theta \leftarrow \theta + \xi_a \nabla_{\theta} \mathcal{J}(\theta) \tag{28}$$

To train the parameters ω of the critic network, the mean square error (MSE) of TD-error can be denoted as a loss function, i.e.,

$$\mathcal{L}(\omega) = \frac{1}{N_c} \sum_t (q(s_t, a_t, g; \omega) - y_t)^2 \tag{29}$$

Algorithm 1: DDPG-Based Framework for RA in OR-BIDT

Setup the max value epoch E_{max}.
Setup the number of time slots in each epoch T.
Setup the capacity in the pool N_c.
Setup learning rates of actor and critic networks ξ_a, ξ_b.
Setup the hyper-parameters of target networks τ.
Initialize parameters θ and ω of networks.
Initialize $\theta^- = \theta$ and $\omega^- = \omega$ of target networks.
Empty the experience replay pool $P_E = \emptyset$.
for $i = 1$ to E_{max} **do**
  Observe the initial state s_1 from the system
  for $i = 1$ to E_{max} **do**
    Generate a goal g.
    Obtain the action a_t from $\pi(s_t, g; \theta)$.
    Take the action a_t.
    Observe the next state s_{t+1} and get r_t^G.
    if $size(P_E)! = FULL$ **then**
      Store $(s_t, a_t, s_{t+1}, r_t^G, g)$ into P_E.
      else
        Randomly select n_0 transactions as P_n from P_E.
        Execute the Algorithm 2, **LSHER**(P_n).
        Update θ with ξ_a according to (28).
        Update ω with ξ_b according to (31).
      end
    end
    Update θ^- and ω^- of target networks with τ according to (32) and (33).
  end
end

The policy gradient of the critic network can be found by the chain derivative rule:

$$\nabla\ {}_{\omega}\mathcal{L}(\omega) = \frac{2}{N_c}\sum_{t}(q_t - y_t)\frac{\partial q(s_t, a_t, g; \omega)}{\partial \omega} \tag{30}$$

Denote the learning rate in the critic network as ξ_b, where $\xi_b \in (0, 1)$, and the parameters can be updated by:

$$.\omega \leftarrow \omega - \xi_b \nabla\ {}_{\omega}\mathcal{L}(\omega) \tag{31}$$

After updating the actor and critic networks, the target networks need to be updated as well. Set a hyperparameter $\tau \in (0, 1)$, parameters θ^- and ω^- can be modified by the soft update algorithm as follows:

$$\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^- \tag{32}$$

$$\omega^- \leftarrow \tau\omega + (1 - \tau)\omega^- \tag{33}$$

The detailed workflow of the presented algorithm for our system is shown in Algorithm 1.

B. LSH-Based Experience Replay Approach

Based on the definition in this paper, it can be found that the reward for GCRL tends to be very sparse. Although the classic hindsight experience replay (HER) algorithm that was proposed in [37] can well address this problem, it often fails to consistently generate the most relevant goals, especially in high-dimensional state spaces characterized by sparse distributions. To overcome this issue, we introduce a novel LSHER algorithm, specifically designed for addressing GA-MDP problems.

Algorithm 2: LSHER-Based Algorithm in Proposed Algorithm

Input: A subset of the experience replay pool P_n.
for tx in P_n **do**
  Retrieve all states from the buckets corresponding to the hash code $\mathcal{H}_l^{hash}(tx)$.
  Form a candidate set $\mathcal{C}_{state}$.
  Select s^+ with the minimum distance from $\mathcal{C}_{state}$.
  Map a new goal $g^+ = \psi(s^+)$.
  $r^* = r^G(tx.state, tx.action, g^+)$.
  Remove tx with $P_n.remove(tx)$.
  $tx = (tx.state, tx.action, tx.next\_state, r^*, g^+)$
  Append tx with $P_n.append(tx)$.
end

Suppose that using the policy to explore in the environment for a goal gives us this one trajectory $\{s_1, s_2, ..., s_T\}$, and $g \notin \{s_1, s_2, ..., s_T\}$. This means that all rewards the agent get in this entire trajectory are -1, which helps very little with the training. Although the goal g was not reached, during its exploration, the agent accomplished equal corresponding goals $\{s_1, s_2, ..., s_T\}$, which meant it accomplished goals $\{\psi(s_1), \psi(s_2), ..., \psi(s_T)\}$. If these goals are utilized to replace the original goal g with a new goal g^- and recalculate the rewards in the trajectory, it allows the policy to derive information useful for network training from experiences of failure. Based on the above idea, firstly, the experience replay pool P_E is initialized, and then the trajectories obtained by sampling the environment with the policy according to the goal g and initial state s_1 are stored in P_E in the form of quintuple (s, a, s^-, r, g). After that, some quintuples (s, a, s^-, r, g) are sample from the pool P_E and a state s^+ is selected to map as a new goal $g^+ = \psi(s^+)$. The new reward r^* is calculated by $r^* = r^G(s, a, g^+)$, and (s, a, s^-, r, g) is replaced by (s, a, s^-, r^*, g^+), which can be used for policy training. Here, we select the nearest future state as the hindsight goal, as it is more likely to be both relevant and beneficial to the learning process. Let s_t be the current state at the time t, and $\{s_{t+1}, s_{t+2}, \ldots, s_T\}$ be the set of future states in the episode. The objective is to select a state s^* from this set such that the distance $d(s_t, s^*)$ is minimized. LSHER leverages LSH to efficiently identify the nearest future state by grouping similar states into the same bucket with high probability, which significantly reduces the computational cost of nearest neighbor searches in high-dimensional spaces. We define the LSH family for Euclidean distance as follows:

$$h_{\mathbf{a},\beta_{rs}}(s_t) = \left\lfloor \frac{\mathbf{a} \cdot s_t + \beta_{rs}}{\delta_{bin}} \right\rfloor \tag{34}$$

where $\mathbf{a}$ is a random vector with each component drawn from a

Gaussian distribution $\mathcal{N}(0,1)^{|s_t|}$, β_{rs} is a random scalar drawn from a uniform distribution $\mathcal{U}(0,\delta_{bin})$ and δ_{bin} is the width of the projection bin. Subsequently, a hash table is created following the steps outlined below.

1) *Multiple Hash Tables:* We define k independent hash functions $h_1, h_2, \ldots, h_k$, each of which is constructed through the equation (34). These hash functions are applied to the state s_t, resulting in a composition of hash values. Then, construct L independent hash tables $\mathbb{H}^{hash} = \{\mathcal{H}_l^{hash} | l = 1, 2, \ldots, L\}$, each with a hash value composition. Formally, this composition is denoted as:

$$\mathcal{H}_l^{hash}(s_t) = (h_1^l(s_t) \,||\, h_2^l(s_t) \,||\, \cdots \,||\, h_k^l(s_t)) \quad (35)$$

2) *Hashing Future States:* For each future state s_i, compute $L \times k$ hash functions to derive its hash codes.
3) *Bucket Insertion:* Insert each state s_t into the corresponding buckets in each of the hash tables.

Once the hash tables have been constructed, the nearest state can be efficiently identified by the procedure outlined in the Algorithm 2.

VI. EXPERIMENTS AND DISCUSSION

In this section, the simulation results of the proposed LSHER-based GCRL algorithm for the OR-BIDT system have been performed.

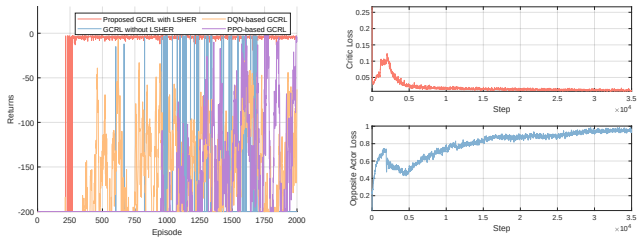
A. Simulation Setup

The experiments were executed in the Microsoft Windows 11 Enterprise operating system (CPU 12th Gen Intel (R) Core (TM) i7-12700KF 3.60 GHz, GPU NVIDIA GeForce RTX 4060 Ti, RAM 32.0 GB). In the simulation, we conducted a OR-BIDT system, in which 1 CS, 5 edge servers and 20 EDs are established for the standard scheme. For the channel gain $g_i^x(t)$ and $G_i^y(t)$, we set channel gain transition probability similar to [11], and divided them into 10 and 8 levels, which are $\left[10^{-5}, 10^{-3},\ 0.2, 0.5, 0.8, 1.1, 1.4, 1.7, 2.0, 2.6\right]$ and $[3.0, 3.3, 3.6, 3.9, 4.2, 4.5, 4.8, 5.1]$, respectively. For a clearer description of the key parameters in the simulation [11], [30], [38], we present them in Table I. To give a further evaluation of the proposed GCRL-based algorithm with LSHER for the OR-BIDT system in RA problem, we provide other three schemes. Since some schemes can only handle discrete action spaces, we discretize the action space using a freedom parameter μ_d for fair comparisons. Specifically, the CPU frequency of each ESN can be calculated by a natural number $n_y(t)$, which means $f_{E,y}(t) = \frac{f_{E,max}}{d} n_y(t)$, $n_y(t) \in \mathbb{N}$ and $\sum_{y \in \mathcal{Y}} n_y(t) = \mu_d$. The actual action space is discretized can be replaced by $a(t) \triangleq \{\mathcal{O}^D(t), \mathcal{O}^E(t), n_1(t), n_2(t), ..., n_{\mathcal{Y}}(t)\}$, so that PPO and DQN algorithms are available for proposed OR-BIDT. The experiment is simulated for 2000 time slots, and the learning losses are shown in 3.5×10^4 steps.

For evaluation of the presented algorithm, following four schemes are considered in the simulation:

TABLE I
KEY PARAMETERS IN THE SIMULATION

Parameter	Value
Uplink transmission bandwidth from EDs to ESNs	20MHz
Uplink transmission bandwidth from ESNs to CS	35MHz
Required CPUS cycles of ESNs	$[0.5, 3.0]$ Gcycles/s
Transmit power	3W
Computing power	2.5W
Each size of one data segment	256KB
Number of data segments of each ED	1 100
Success density moderation factor of ESNs	0.8
Number of transactions in each block	100
Learning rate of actor network	0.0001
Learning rate of critic network	0.0003
Weight factors in consensus process	$\{0.3, 0.7\}$
Soft update parameter for target networks	0.4



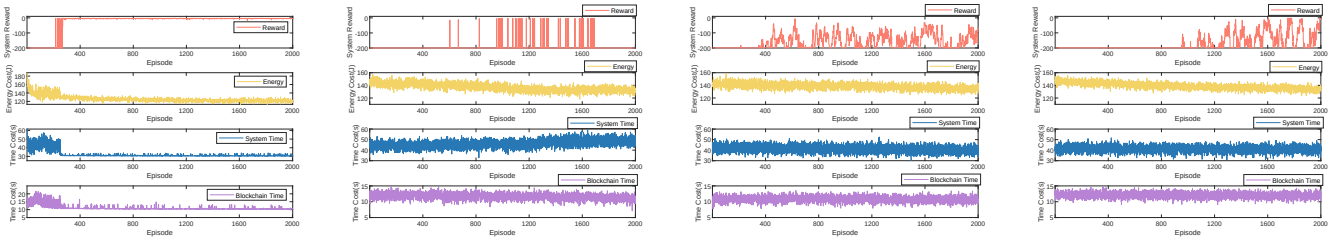
(a) (b)

Fig. 4. Experimental results of different schemes for OR-BDT, including (a) Returns of different algorithms, (b) Learning Loss of Proposed LSHER-based DDPG for OR-BDT.

1) **GDWL** *(DDPG-based scheme of GCRL with LSHER)*: This scheme is the proposed scheme, which adopts DDPG with LSHER to address the issue of sparse returns.
2) **GDWOL** *(DDPG-based scheme of GCRL without LSHER)*: This scheme is the same as our proposed scheme, except that the LSHER algorithm is not utilized.
3) **GPPO** *(PPO-based scheme of GCRL)*: This scheme uses PPO-based algorithms to solve the problem, and the action space is discrete.
4) **GDQN** *(DQN-based scheme of GCRL)*: This scheme apples DQN-based algorithms to solve the problem, and the action space is discrete.

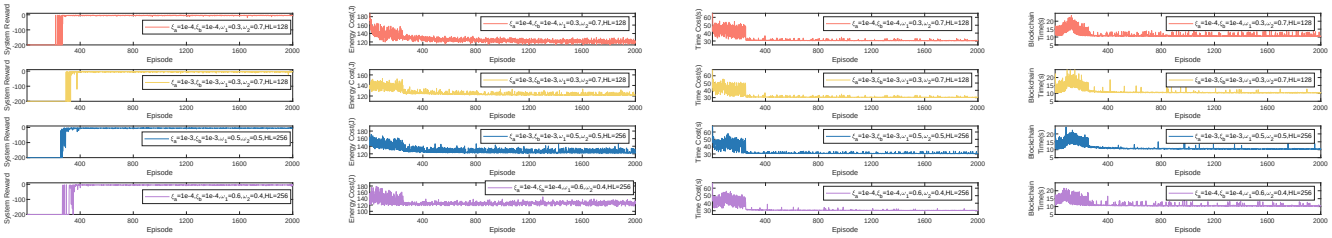
B. Convergence Performance

We first show the total returns of the proposed LSHER-based DDPG algorithm, DDPG without HER algorithm, DQN-based algorithm, and PPO-based algorithm for OR-BDT in Fig. 4 (a). In this simulation, there are 5 edge servers, 20 EDs and 1 CS. The figure illustrates that our LSHER-based DDPG algorithm has the best convergence and obtains the maximum returns compared with the other three schemes. In the meanwhile, the curve of DWL means that the algorithm can stabilize the system in a shorter period of convergent time and maintain sustained returns. It is worth noting that none of the other three schemes converge well within the specified episode. This is because that DWOL scheme does not use the HER algorithm, which ultimately fails to converge due



(a) GDWL (b) GDWOL (c) GPPO (d) GDQN

Fig. 5. Experimental results of different schemes for OR-BIDT, including (a) LSHER-based GCRL, (b) GCRL scheme without LSHER for OR-BIDT, (c) DQN-based scheme and (d) PPO-based scheme.



(a) Returns (b) System time (c) System energy (d) Blockchain time

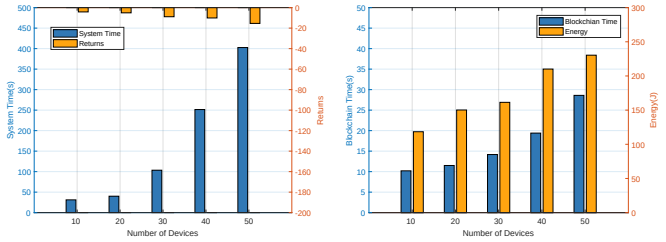
Fig. 6. Performance of GDWL with different learning rates, weights, and hidden layers in neural networks for OR-BIDT, including (a) returns, (b) system time cost, (c) system energy cost and (d) blockchain time cost.

to the sparsity of returns. In PB and DB schemes, the action space becomes a discrete space and does not converge well, since there is the curse of dimensionality. With the increase of iterations, DWL is the only method that reaches a convergence and maximizes returns. This indicates that the proposed DWL scheme for OR-BDT is more powerful and functional than the other three methods, since our algorithm can make better real-time decisions in DT. However, the convergence of the returns of the system from negative to 0 in the Fig. 4 (a) is attributed to the fact that OR-BDT employs the GCRL model, where the distance of the state space mapping to the goal is less than a given value, and the reward is 0, otherwise it is -1. Fig. 4 (b) demonstrates that the convergence curves of the learning loss for the actor network and the critic network. Note that the critic network needs to minimize the loss function MSE, so the learning loss is decreasing and converge at about 1×10^4 steps. While the actor network needs to maximize the Q-value (minimize the opposite of Q-value), the opposite of the learning loss is increasing and converge at almost 3×10^4 steps.

C. Performance of the Proposed Algorithm

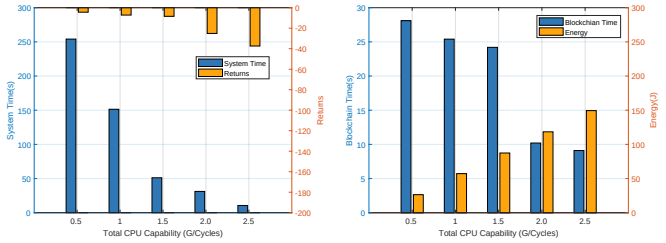
Fig. 5 compares the performance of different OR-BIDT schemes. The LSHER-based DDPG algorithm consistently achieves the designated goals, maintaining returns at 0. This leads to the convergence of energy cost, system time, and blockchain time at approximately 120, 30, and 10, respectively—lower and more stable than those of other schemes (b), (c), and (d). These results demonstrate that the GCRL-based algorithm of DDPG with LSHER algorithm outperforms GDWOL, GPPO, and GDQN by effectively reducing energy consumption, time costs, and blockchain latency while ensuring system security. Moreover, it supports real-time offloading and resource allocation, adapting to varying DT model requirements, such as minimizing time or energy costs.

Fig. 6 shows the total system returns, system time cost, energy cost and blockchain time cost of the proposed LSHER-based DDPG algorithm with different parameters. Here, we adopt learning rates, weight factors of holding stakes, and hidden layers in the algorithm, which includes $(\xi_a = 10^{-4}, \xi_b = 10^{-4}, \omega_1 = 0.3, \omega_2 = 0.7, HL = 128)$, $(\xi_a = 10^{-3}, \xi_b = 10^{-3}, \omega_1 = 0.3, \omega_2 = 0.7, HL = 128)$, $(\xi_a = 10^{-3}, \xi_b = 10^{-3}, \omega_1 = 0.5, \omega_2 = 0.5, HL = 256)$, $(\xi_a = 10^{-4}, \xi_b = 10^{-4}, \omega_1 = 0.6, \omega_2 = 0.4, HL = 256)$. The algorithm can converge well in these four options. When the parameters are set to $(\xi_a = 10^{-4}, \xi_b = 10^{-4}, \omega_1 = 0.3, \omega_2 = 0.7, HL = 128)$, the system return converges fastest compared with other three options, and the energy cost, system time cost and blockchain time cost are gradually reduced until they stabilize. This is because settled hidden layers and learning rates enable neural networks to fit faster functional relationships between the physical and digital worlds. In the meantime, $\omega_1 = 0.3$ and $\omega_2 = 0.7$ allow ESNs to vote more efficiently with holding stakes. When the parameters are set to $(\xi_a = 10^{-3}, \xi_b = 10^{-3}, \omega_1 = 0.3, \omega_2 = 0.7, HL = 128)$, as the learning rates increase and the neural network contains fewer nodes in the hidden layer, the system return has the slowest convergence, but the time and energy cost are still decreasing, which illustrates the effectiveness of OR-BDT. When the parameters are set to $(\xi_a = 10^{-3}, \xi_b = 10^{-3}, \omega_1 = 0.5, \omega_2 = 0.5, HL = 256)$, the energy consumption curve fluctuates a bit before convergence, due to the variation of the



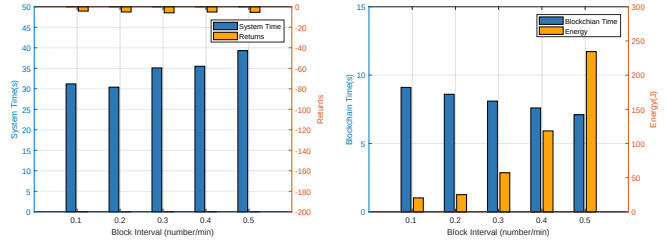
(a) (b)

Fig. 7. Performance of LSHER-based GCRL algorithm with different number of devices, including (a) system time cost and returns, (b) blockchain time cost and system energy cost.



(a) (b)

Fig. 8. Performance of LSHER-based GCRL algorithm with different total CPU capability, including (a) system time cost and returns, (b) blockchain time cost and system energy cost.



(a) (b)

Fig. 9. Performance of LSHER-based GCRL algorithm with different block intervals, including (a) system time cost and returns, (b) blockchain time cost and system energy cost.

weight factors ω_1 and ω_2. Similarly, the system returns and energy consumption fluctuate when $\omega_1 = 0.6$ and $\omega_2 = 0.4$, but the rate of convergence is slightly different, since there are some modifications in the learning rate and the nodes in the hidden layer. Therefore, we can conclude that the proposed LSHER-based DDPG algorithm is adapted to various complex scenarios and makes the system meet the requirement of the consumption of less energy or lower latency.

Fig. 7 shows illustrates the performance of LSHER-based GCRL algorithm with different number of devices in four aspects: returns, system time cost, blockchain time cost and energy consumption. To better assess the impact of variable numbers of devices on these four measures, we assigned five levels to the number of devices, $10, 20, 30, 40, 50$. Experimental comparisons with different numbers of devices are of great importance for practical scenarios. From Fig. 7 (a), we can observe that as the number of devices grows, the system returns fluctuate and the time consumed by the system increases, especially after the number of devices exceeds 30. Similarly, in Fig. 7 (b), it can be shown that the time consumed by the blockchain module and the energy consumption of the system increases. On the one hand, the returns tend to decrease because the system converges more slowly as the number of devices increases. On the other hand, the time consumed by the blockchain, and the system energy consumption mainly come from the edge servers, but the time consumed by the system mainly comes from the number of devices. Therefore, the difference between the maximum and minimum of the two evaluation criteria in Fig. 7 (b) is not as large as the difference of the system time in Fig. 7 (a).

Fig. 8 demonstrates the performance of LSHER-based GCRL algorithm with different total CPU capability. Here, we set the total CPU capacity (G/Cycles) of edge servers to five levels, and they are 0.5, 1.0, 1.5, 2.0, 2.5, respectively. In Fig. 8 (a), the system converges more slowly as the total capacity of the edge server increases, and the system time is significantly reduced. In Fig. 8 (b), the consumption time of the blockchain module is the same as the system time, but the energy consumption increases dramatically. This is because the larger the total capacity becomes, the more difficult it is for the system to allocate resources to each server, and then the slower the system converges.

Fig. 9 shows the performance of LSHER-based GCRL algorithm with different block intervals. We set the block interval (number/minute) to 0.1, 0.2, 0.3, 0.4, and 0.5, which means 1, 2, 3, 4, 5 blocks are produced by ESNs every 10 minutes. In Fig. 9 (a), we can observe that the system returns are almost unchanged when the block interval is varied within a given range. There is only a small increase in system time cost, which is due to the fact that more blocks produced per unit of time take up more resources of edge servers. In Fig. 9 (b), when blocks are generated faster, the more energy is consumed while the less blockchain time is spent, since edge servers utilize more CPU resources and blockchain time cost consists of block generation time, broadcast time and validation time.

VII. CONCLUSION AND FUTURE WORK

In this article, we proposed the OR-BIDT to analysis statistics in the digital world and guide decisions in the physical world. To achieve the required performance of the system, we presented a hierarchical offloading mechanism and defined the goal space according to time delay, energy consumption, block interval, and block delay, which was formulated as a GA-MDP. Then, we developed a GCRL algorithm with LSHER-based DDPG to address the problem of playing back high-dimensional experiences. Experimental results showed that our algorithm converged very quickly with different parameters. In future work, we will consider interference management and goal-conditioned multi-agent (GC-MA) algorithms in blockchain-based systems.

REFERENCES

[1] J. Kang, Z. Xiong, X. Li, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Optimizing task assignment for reliable blockchain-empowered federated edge learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1910–1923, 2021.

[2] M. Ibrar, L. Wang, A. Akbar, M. A. Jan, V. Balasubramanian, G.-M. Muntean, and N. Shah, "Adaptive capacity task offloading in multi-hop d2d-based social industrial iot," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2843–2852, 2022.

[3] D. Feng, L. Lu, Y. Yuan-Wu, G. Y. Li, S. Li, and G. Feng, "Device-to-device communications in cellular networks," *IEEE Communications Magazine*, vol. 52, no. 4, pp. 49–55, 2014.

[4] Y. Su, J. Li, J. Li, Z. Su, W. Meng, H. Yin, and R. Lu, "Robust and lightweight data aggregation with histogram estimation in edge-cloud systems," *IEEE Transactions on Network Science and Engineering*, 2024.

[5] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 1387–1395.

[6] T. Wang, B. Sun, L. Wang, X. Zheng, and W. Jia, "Eidls: An edge-intelligence-based distributed learning system over internet of things," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[7] 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 of Things Journal*, vol. 8, no. 4, pp. 2276–2288, 2020.

[8] J. Leng, X. Zhu, Z. Huang, K. Xu, Z. Liu, Q. Liu, and X. Chen, "Manuchain ii: Blockchained smart contract system as the digital twin of decentralized autonomous manufacturing toward resilience in industry 5.0," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[9] C. Avasalcai, C. Tsigkanos, and S. Dustdar, "Resource management for latency-sensitive iot applications with satisfiability," *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2982–2993, 2021.

[10] L. Cai, Y. Dai, Q. Hu, J. Zhou, Y. Zhang, and T. Jiang, "Bayesian game-driven incentive mechanism for blockchain-enabled secure federated learning in 6 g wireless networks," *IEEE Transactions on Network Science and Engineering*, 2024.

[11] D. Wang, B. Li, B. Song, Y. Liu, K. Muhammad, and X. Zhou, "Dual-driven resource management for sustainable computing in the blockchain-supported digital twin iot," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6549–6560, 2022.

[12] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *Ieee Access*, vol. 8, pp. 21 980–22 012, 2020.

[13] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[14] K. Zhang, J. Cao, S. Maharjan, and Y. Zhang, "Digital twin empowered content caching in social-aware vehicular edge networks," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 239–251, 2021.

[15] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5g services in mobile edge computing systems: Learn from a digital twin," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4692–4707, 2019.

[16] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5g beyond," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 175–12 186, 2020.

[17] S. Sharma, N. Gupta, and V. A. Bohara, "Insights from the measurement results for adaptive and cooperative d2d communication framework," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[18] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng, "Qos-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 4027–4041, 2019.

[19] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4188–4200, 2018.

[20] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2018.

[21] Z. Lin, S. Bi, and Y.-J. A. Zhang, "Optimizing ai service placement and resource allocation in mobile edge intelligence systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7257–7271, 2021.

[22] Y. Zhang, Z. Feng, H. Moustafa, F. Ye, U. Javaid, and C. Cui, "Guest editorial: Edge intelligence for beyond 5g networks," *IEEE Wireless Communications*, vol. 28, no. 2, pp. 10–11, 2021.

[23] Z. Zhao, R. Zhao, J. Xia, X. Lei, D. Li, C. Yuen, and L. Fan, "A novel framework of three-hierarchical offloading optimization for mec in industrial iot networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5424–5434, 2019.

[24] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 695–708, 2018.

[25] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE internet of things journal*, vol. 6, no. 3, pp. 4660–4670, 2018.

[26] J. Xu, S. Wang, B. K. Bhargava, and F. Yang, "A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3538–3547, 2019.

[27] M. Xevgenis, D. G. Kogias, P. Karkazis, H. C. Leligou, and C. Patrikakis, "Application of blockchain technology in dynamic resource management of next generation networks," *Information*, vol. 11, no. 12, p. 570, 2020.

[28] Y. Wang, C.-R. Chen, P.-Q. Huang, and K. Wang, "A new differential evolution algorithm for joint mining decision and resource allocation in a mec-enabled wireless blockchain network," *Computers & Industrial Engineering*, vol. 155, p. 107186, 2021.

[29] J. Feng, F. R. Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6214–6228, 2019.

[30] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: A deep reinforcement learning approach," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3559–3570, 2019.

[31] S. Fei, Z. Yan, H. Xie, and G. Liu, "Sec-e2e: End-to-end communication security in ls-hetnets based on blockchain," *IEEE Transactions on Network Science and Engineering*, 2023.

[32] D. Larimer, "Dpos consensus algorithm-the missing white paper," *Bitshare whitepaper*, 2017.

[33] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, 2008.

[34] M. Simsek, M. Bennis, and I. Güvenç, "Learning based frequency- and time-domain inter-cell interference coordination in hetnets," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4589–4602, 2014.

[35] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in hetnets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680–692, 2017.

[36] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," *arXiv preprint arXiv:2201.08299*, 2022.

[37] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[38] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-efficient federated learning for digital twin edge networks in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5709–5718, 2020.