Hindawi
Mathematical Problems in Engineering
Volume 2023, Article ID 8858263, 11 pages
https://doi.org/10.1155/2023/8858263

Research Article

Design and Optimization in MEC-Based Intelligent Rail System by Integration of Distributed Multi-Hop Communication and Blockchain

Linlin Tian,[1] **Meng Li**,[1,2] **Pengbo Si**,[1,2] **Ruizhe Yang**,[1,2] **Yang Sun**,[1] **and Zhuwei Wang**[1]

[1]*Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China*
[2]*Beijing Laboratory of Advanced Information Networks, Beijing 100124, China*

Correspondence should be addressed to Meng Li; limeng720@bjut.edu.cn

Received 15 August 2022; Revised 17 October 2022; Accepted 24 November 2022; Published 3 February 2023

Academic Editor: Li Zhu

Mobile edge computing technology has emerged as a novel computing paradigm that makes use of resources close to the devices of the smart rail system. Nevertheless, it is difficult to support data offloading to the stations directly from different trains due to the limited coverage of the stations equipped with MEC servers. Therefore, multi-hop ad hoc network is considered and introduced in this case. In this paper, an improved architecture is proposed for the MEC-based smart rail system by blockchain and multi-hop data communication. The requesting trains can offload the tasks to MEC servers by multi-hop transmission between trains, even when requesting trains are not covered by servers. Furthermore, we utilize the blockchain technology for the authenticity and anti-falsification of information during multi-hop transmission. Then, the offloading routing path and offloading strategy are co-optimized to minimize both delay and cost of the system. The proposed majorization problem is formulated as a Markov decision process (MDP) and solved by deep reinforcement learning (DRL). In comparison to other existing schemes, simulation results demonstrate that the proposed scheme can greatly improve system performance.

1. Introduction

As the smart rail system continues to grow, it is urgent to realize the dynamic aggregation, deep mining, and effective utilization of various application data by building high-performance ubiquitous computing power. Cloud computing was employed to resolve the issue, which is because of the constrained processing power of the trains [1]. However, it is obvious that cloud computing architecture cannot meet the real-time requirements for information processing in the smart rail system, on account of the rapid mobility of trains [2, 3]. Fortunately, mobile edge computing (MEC), as an emerging technology, solves the issue mentioned above effectively. Meanwhile, the MEC technology performs computing tasks on edge servers close to the device rather than on the cloud, which meets the sensitive delay requirements. At the same time, it brings high-quality services to users [4, 5].

However, the coverage of the stations equipped with MEC servers is limited, so it is impractical to only consider that the trains are within the range of the MEC servers. Multi-hop ad hoc network has no fixed topology. Train nodes can spontaneously create wireless network for communication between trains to exchange information and data. Each train is not only a transceiver but also a router [6]. Therefore, we consider integrating the multi-hop ad hoc network and MEC technology. The requesting trains can offload the tasks to MEC servers by multi-hop transmission between trains, which enables the servers to be utilized in a wider range while meeting the low-latency requirement.

Although the combination of multi-hop ad hoc network and MEC in the smart rail system can bring great

advantages, since a large amount of information related to traffic and driving is involved, how to effectively guarantee the security and reliability during data multi-hop transmission is worth considering. Fortunately, due to the distributed, immutable, and safety nature of blockchain, blockchain is applicable to prevent the information related to traffic and driving from being leaked or manipulated for the MEC-enabled smart rail system with multi-hop connection [7–11].

However, there are still significant obstacles to overcome before multi-hop ad hoc and blockchain can be effectively applied in the MEC-enabled smart rail system. For instance, how to properly select the routing path and the offloading decision with the high-speed movement of trains is deemed as the crucial problem. In addition, how to balance the delay and cost caused by the process of data delivery, offloading, and consensus in the MEC-enabled smart rail system also needs to be considered.

In this paper, to deal with the mentioned issues, we propose an improved optimization framework for the MEC-enabled smart rail system by multi-hop data communication and blockchain. Then, the offloading routing path, offloading strategy, and block size are co-optimized to minimize both delay and cost of the system during communication and computation process. Furthermore, by specifying the state space, action space, and reward function, a discrete Markov decision process (MDP) is formulated to characterize the dynamic jointly proposed problem. Additionally, we utilize dueling deep Q-learning network (DQN) for obtaining the optimal strategy.

The rest of this paper is structured as follows. Section 2 mainly proposes the system model. Then, we formulate the collaborative majorization problem in Section 3. In Section 4, the formulated problem is solved by dueling DQN algorithm. The experiment results are presented and discussed in Section 5. The last part summarizes the conclusion of this paper and the future directions.

2. System Model

In this section, we depict the system model, which consists of the network model, multi-hop routing path model, communication model, computation model, and blockchain model.

2.1. Network Model. In Figure 1, the architecture of a high-speed railway and a train station equipped with MEC servers which are managed by various suppliers is shown. The available computing resource and price of each MEC server are different in a real-time environment. We denote the set of these MEC servers as $R = \{1, \ldots, r, \ldots, R\}$. There are several high-speed trains running on the tracks. We denote $V = \{1, \ldots, v, \ldots, V\}$ as the set of all trains. Multi-hop ad hoc network is utilized to assist requesting trains in computation task offloading. Trains can act as relaying nodes to realize information interaction with other trains by spontaneously creating wireless network.

There may be malicious relaying nodes when offloading computation tasks by relaying. Therefore, the trust-based blockchain system is utilized to ensure the authenticity and anti-falsification of data information during the relaying and offloading process. The last-hop relaying train sends the data consensus requirement and transaction information to the blockchain system for transaction verification after receiving the relaying task. Through the consensus mechanism, the requesting train node and the other relaying train nodes in the routing path check the information data. In the blockchain system, all trains are regarded as blockchain nodes. These nodes can play either a normal or a consensus node role. Normal trains are in charge of transferring and accepting ledger information, while consensus trains are in charge of creating new blocks and carrying out the consensus process. Each relaying train in the routing path is regarded as a candidate for consensus nodes, and we consider the trust value of each relaying train when voting for consensus.

In this work, the requesting trains can maximize the use of multi-hop ad hoc network to offload tasks to the MEC servers, even if the trains are not in the communication range of servers. A consensus process is initiated when the last relaying train receives the offloading task, and the security of information is guaranteed when all consensus nodes reach a consensus successfully. For each task, there are two important elements to be considered: latency and cost. In terms of latency, considering the link quality along the whole routing path and the processing capability of servers, the total expectable latency of one successful end-to-end transfer and calculation is evaluated. In terms of cost, the total expected cost is assessed, including the data relaying cost of each relaying train in the whole path and computing cost of different MEC servers. As a result, we can select the optimal routing path and offloading decision which has the minimal latency and cost.

2.2. Multi-Hop Routing Path Model. Firstly, to determine the performance of each pair of trains in multi-hop routing path, we depict a link model, which considers channel fading and mobility of trains. Then, based on the link quality obtained above, the routing metric about link correlation is utilized to select the optimal multi-hop routing path.

2.2.1. Link Quality. We utilize the Nakagami distribution model to represent the fading of radio wave propagation [12]. Thus, the successful delivery probability between sender train v_i and receiver train v_j in spite of channel fading can be obtained by

$$p_{ij}^{f}(t) = 1 - F_d(r_{rt}; m, \varphi), \tag{1}$$

where $F_d(r_{rt}; m, \varphi)$ is the cumulative distribution function of the receiving signal power, r_{rt} is the reception threshold of a signal, and φ is the average signal strength. The fading parameter m is related to distance $D(v_i, v_j, t)$ between train v_i and train v_j at current time t as follows:

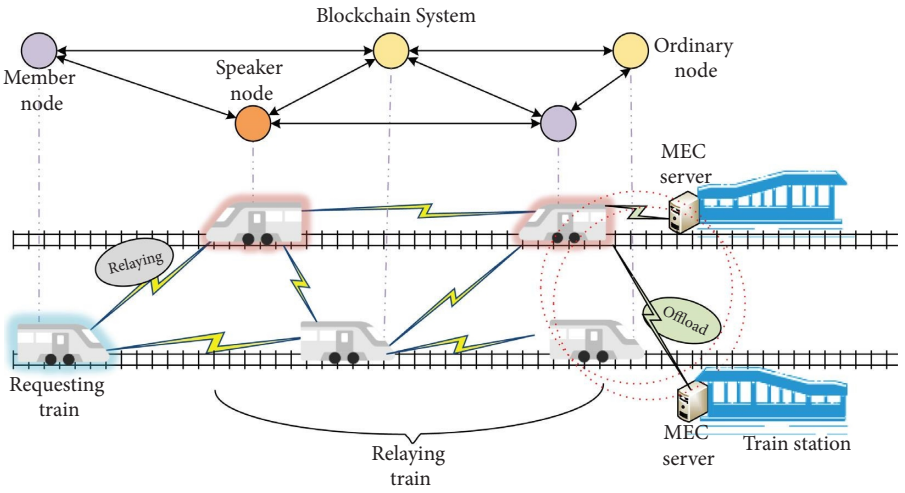

Figure 1: Structure of the proposed network scenario.

$$m\left(D\left(V_i, V_j, t\right)\right) = \begin{cases} 1, D\left(V_i, V_j, t\right) \geq 150m, \\ 1.5, 50m \leq D\left(V_i, V_j, t\right) < 150m, \\ 3, D\left(V_i, V_j, t\right) < 50m. \end{cases} \quad (2)$$

Under a real urban environment, the train does not always run at a constant speed, and its speed changes with acceleration or deceleration. In this case, the movement of the trains can be abstracted by a Wiener process [13]. Assuming that the trains only have two directions toward each end, one direction is designated as positive. Therefore, the velocity variation of train v_i during interval $[t_0, t]$ can be defined as follows:

$$\Delta v_i^{t-t_0} = v_i^t - v_i^{t_0} = \mu_i\left(t - t_0\right) + \delta_i\sqrt{t - t_0}, \quad (3)$$

where $v_i^{t_0}$ and v_i^t represent the velocity of train v_i at time t_0 and t. The drift parameter μ_i denotes the acceleration or deceleration of train v_i, and the parameter δ_i follows the Gaussian distribution. Therefore, the relative distance variation of train v_i and v_j in period $[t_0, t]$ can be obtained as

$$\Delta D\left(V_i, V_j, t\right) = \left(v_i^{t_0} - v_j^{t_0} + \mu_i - \mu_j + \Delta v_i^{t-t_0} - \Delta v_j^{t-t_0}\right) * \left(t - t_0\right), \quad (4)$$

and $D(v_i, v_j, t)$ is the distance between two trains at current time t under different circumstances, which are presented as follows.

Circumstance 1: two trains are moving in one direction:

$$D\left(V_i, V_j, t\right) = \begin{cases} \Delta D\left(V_i, V_j, t\right) + d_{ij}, \text{if } v_i > v_j, \text{train } V_i \text{ is in front of train } V_j, \\ \Delta D\left(V_i, V_j, t\right) - d_{ij}, \text{if } v_i > v_j, \text{train } V_i \textit{ is} \text{ behind train } V_j. \end{cases} \quad (5)$$

Circumstance 2: two trains are running in the opposite direction:

$$D\left(V_i, V_j, t\right) = \begin{cases} \Delta D\left(V_i, V_j, t\right) - d_{ij}, \text{two trains are dr iving towards each other}, \\ \Delta D\left(V_i, V_j, t\right) + d_{ij}, \text{two trains are dr iving away from each other}, \end{cases} \quad (6)$$

where d_{ij} denotes the distance between two trains at time t_0. Thus, we can predict link availability and obtain the probability of link availability on link L_{ij} as

$$P_{ij}^{\ a}(t) = P\left(D\left(V_i, V_j, t\right)\right) <= R, \quad (7)$$

where R is the communication range of trains. Therefore, according to equations (1) and (7), we can

obtain the link quality of link L_{ij}, which represents the probability of successful transmission of the packets and is calculated by

$$p_{ij}(t) = p_{ij}^{f}(t) \times p_{ij}^{a}(t). \tag{8}$$

2.2.2. Routing Path Quality. Once data packet loss occurs on one link, this packet should be retransmitted from the source to the destination, on account of the retransmission mechanism in the transport layer. As a result, there are various retransmission times and consumption of network resources for packets in different multi-hop paths. This phenomenon is referred to as a link correlation. Thus, the expected retransmission probability $Fr_N(t)$ of the N-hop path can be calculated by

$$Fr_N(t) = \left(1 - p_{s1}(t)\right) + 2 \cdot p_{s1}(t) \cdot \left(1 - p_{12}\right) + \ldots + N \cdot p_{s1}(t) \cdot p_{12}(t) \cdots \left(1 - p_{(N-1)N}(t)\right), \tag{9}$$

where $p_{s1}(t)$ is the link quality between the source train and the first relaying train and $p_{(N-1)N}(t)$ represents link quality between the relaying train $N-1$ and relaying train N in the routing path.

Therefore, we define the expected times of data transfer in an N-hop routing path as $Fc_N(t)$ when one packet is successfully transferred from source to destination and present it as follows:

$$Fc_N(t) = \frac{Fr_N(t) + N \cdot p_{s1}(t) \cdot \prod_{i=2}^{N} p_{(i-1)i}(t)}{p_{s1}(t) \cdot \prod_{i=2}^{N} p_{(i-1)i}(t)}, \tag{10}$$

where $p_{s1}(t) \bullet \prod_{i=2}^{N} p_{(i-1)i}(t)$ is the aggregation of the link quality of all links in an N-hop routing path.

2.3. Communication Model. In this section, we describe the communication process of the system, including the representation of the communication latency and relaying cost.

2.3.1. Communication Delay. The communication delay is mainly caused by three parts, including the requesting train offloading the task to the last-hop relaying train through the multi-hop ad hoc network, the last-hop relaying train initiating a consensus after receiving the task, and the last-hop relaying train offloading the task to MEC server.

Firstly, the requesting train offloads the task to the last-hop relaying train through the multi-hop ad hoc network. The set of relaying trains in the routing path for the task $I_{MV}(t)$ is denoted as $N = \{1, \ldots, n, \ldots, N\}$ (except the requesting train) and $N \in V$. The data transmission rate per hop communication between relaying train v_i and v_j is obtained as follows:

$$r_{ij}(t) = W \cdot log_2\left(1 + \frac{P_t}{D\left(V_i, V_j, t\right)^h \cdot P_n}\right), \tag{11}$$

where W represents the channel bandwidth, P_t is the transmit power, h indicates the path-loss exponent, and P_n is the background noise power. Thus, the delivery delay of task $I_{MV}(t)$ in V2V N-hop connections can be expressed as

$$T_{\text{tran},V2V}(t) = Fc_N(t) \times \left[\frac{B_{MV}(t)}{r_{s1}(t)} + \sum_{n=2}^{N} \frac{B_{MV}(t)}{r_{(n-1)n}(t)}\right], \tag{12}$$

where $B_{MV}(t)$ denotes input data size required by task $I_{MV}(t)$, $r_{s1}(t)$ is the transmission rate between the source train (i.e., requesting train) and the first relaying train, and $r_{(n-1)n}(t)$ is the transmission rate between the relaying trains v_{n-1} and v_n.

Secondly, the last-hop relaying train sends the data to the blockchain system for transaction verification after receiving the relaying task, so as to guarantee that the data are true without tampering. The delay generated by consensus process is defined as $T_{bc}(t)$, which will be described in detail in the blockchain model.

Finally, the last-hop relaying train offloads the task to the MEC server managed by different suppliers through wireless communication. The Shannon–Hartley theory is used to estimate the uplink rate for data transmission from the last-hop relaying trains to MEC server via LTE cellular network, and it can be calculated as

$$R_{r(r\in\mathcal{R})}(t) = B \cdot log_2\left(1 + \frac{p_{V2I}(t) \times g_{i,r(i\in V)}^{c}}{\Psi^2 + \sum_{j\neq i, j\in V} p_{V2I}(t) \times g_{j,r}^{c}}\right), \tag{13}$$

where B represents the channel bandwidth, Ψ^2 represents the background noise power, $p_{V2I}(t)$ denotes the transmission power of the train (all trains have the same transmitting power), and $g_{i,r(i\in V)}^{c}$ is the channel gain between the train user v_i and MEC server r.

Therefore, the transmission delay caused in this process is calculated as

$$T_{\text{tran},V2I}(t) = \frac{B_{MV}(t)}{R_r(t)}. \tag{14}$$

As mentioned above, we can obtain the total delay of communication process through

$$T_{\text{tran}}(t) = T_{\text{tran},V2V}(t) + T_{bc}(t) + T_{\text{tran},V2I}(t). \tag{15}$$

2.3.2. Communication Cost. We assume that each train has its own relaying price [14]. Corresponding to the relaying trains in the routing path for the task $I_{MV}(t)$, the relaying

price (per unit data volume) sequence is $\{\mathrm{Pr}_1(t), \mathrm{Pr}_2(t), \ldots, \mathrm{Pr}_n(t), \ldots, \mathrm{Pr}_N(t)\}$. Therefore, the total train relaying cost can be obtained by

$$Pr_{\mathrm{sum}}(t) = \sum_{n=1}^{N} Pr_n(t) \times B_{MV}(t). \tag{16}$$

2.4. Computation Model. Assume that each MEC server operated by different suppliers has its corresponding processing capacity and price for computing offloading task. The computing capacity and price (per unit task complexity) of MEC server are $F_{r(r\in)}(t)$ and $Pc_{r(r\in)}(t)$, respectively. Then, the calculation delay and cost are presented as

$$T_{\mathrm{cal}}(t) = \frac{C_{MV}(t)}{F_r(t)}, r \in \mathscr{R}, \tag{17}$$

and

$$Pc_{\mathrm{sum}}(t) = Pc_r(t) \times C_{MV}(t), r \in \mathscr{R}, \tag{18}$$

where $C_{\mathrm{MV}}(t)$ denotes the required CPU cycles for task $I_{\mathrm{MV}}(t)$.

Thus, the total delay and cost, including offloading delivery process and calculation process, are represented by

$$T_{\mathrm{sum}}(t) = T_{\mathrm{tran}}(t) + T_{\mathrm{cal}}(t) = T_{\mathrm{tran},V2V}(t) + T_{bc}(t) + T_{\mathrm{tran},V2I}(t) + T_{\mathrm{cal}}(t), \tag{19}$$

and

$$P_{\mathrm{sum}}(t) = Pr_{\mathrm{sum}}(t) + Pc_{\mathrm{sum}}(t). \tag{20}$$

2.5. Blockchain Model. In this paper, the delegated Byzantine fault tolerance (dBFT) consensus mechanism is adopted in our blockchain system to increase the efficiency of a consensus process without tampering [15]. Moreover, each relaying train in the routing path is regarded as a candidate for consensus nodes, and we consider the trust value of candidates to determine the nodes participating in the next round of consensus, which improves the throughput of blockchain, reduces the CPU cycles of transaction confirmation, and then effectively reduces the consensus latency [16]. The higher the trust value of the relaying node is, the more likely it is to be selected as a consensus node. The set of the selected consensus nodes is denoted by $K = \{1, 2, \ldots, k, \ldots, K\}$ and $K \in N$. The dBFT consensus protocol can be adopted in the proposed system model to dynamically adapt to the change of the number of train nodes [13].

2.5.1. Calculation of Trust Value. Generally, the trust value is determined by its direct trust value and indirect trust value [17, 18]. The trust value of the train node $v (v \in V)$ is defined as D_v^{trust} and $D_v^{\mathrm{trust}} \in [0, 1]$. Similar to [19], the threshold of trust value is set as 0.5. One node is trustworthy to be a candidate for consensus only if its trust value is higher than 0.5.

We utilize subjective logic to compute direct trust value of the blockchain nodes, which can be obtained as follows:

$$D^{\mathrm{direct}}_{i(i\in V, i\neq v)\longrightarrow v} = \begin{cases} 0.5 + \left(NH_{i\longrightarrow v} - 0.5\right) \times NC_{i\longrightarrow v}, \text{if } NH_{i\longrightarrow v} \geq 0.5, \\ NH_{i\longrightarrow v} \times NC_{i\longrightarrow v}, \text{otherwise}, \end{cases} \tag{21}$$

where $\mathrm{NH}_{i\longrightarrow v}(t)$ is the node honesty (NH) and represents the uncertainty during offloading due to unstable and noisy communication channels between the relaying trains [18]. $\mathrm{NC}_{i\longrightarrow v}(t)$ is the remaining node capacity (NC) of the trains to complete task.

For the computation of indirect trust value, the number of times one node has been voted for consensus in the past is taken into account. The blockchain system regularly updates and records the selection of consensus nodes. Thus, the indirect trust value of one blockchain node can be defined as

$$D_v^{\mathrm{indirect}} = \frac{\mathrm{VN}_v}{\mathrm{VN}_{\mathrm{all}}}, \tag{22}$$

where $\mathrm{VN}_{\mathrm{all}}$ is the total number of consensus processes and VN_v is the number of times the relaying train v has been voted for consensus.

Therefore, the trust value of one candidate for consensus is represented by

$$D_v^{\mathrm{trust}} = w_{\mathrm{direct}} \cdot \frac{\sum_{i=1}^{v-1} D^{\mathrm{direct}}_{i\longrightarrow v} + \sum_{i=v+1}^{V} D^{\mathrm{direct}}_{i\longrightarrow v}}{V} + w_{\mathrm{indirect}} \cdot D_v^{\mathrm{indirect}}, \tag{23}$$

where w_{direct} and w_{indirect} represent the weight of direct and indirect trust values, respectively. Meanwhile, w_{direct} and $w_{\text{indirect}} \in [0, 1]$ and $w_{\text{direct}} + w_{\text{direct}} = 1$.

2.5.2. Consensus Process. The specific dBFT consensus process is depicted in Figure 2. We assume that generating/validating one signature and message authentication code (MAC) requires α and β CPU cycles, respectively.

At first, the last-hop relaying train sends consensus requirement and transaction information to blockchain system upon receiving the offloading task. Then, the speaker of the consensus process in this round is assigned by blockchain. The assigned speaker packages the hash of the transactions as a prepare request message to launch a proposal and broadcast it to initiate a new consensus. During this phase, one signature and $K-1$ MACs are generated by the speaker. Thus, the computation cycles for the speaker node in this process are represented as

$$c_1(t) = \alpha + (K-1)\cdot\beta. \tag{24}$$

Secondly, each member collects all the transactions information of the prepare request message. If the transactions are verified successfully, the members add the transactions to the consensus module and broadcast the prepare response messages to all consensus nodes. During this phase, members need to validate the signatures and MACs of the proposal and contained transactions and then generate one signature and $K-1$ MACs for forming prepare response messages. Therefore, the computation cycles for the member nodes are calculated by

$$c_2(t) = \alpha + \beta + \frac{S_{\text{bc}}(t)}{L}\cdot(\alpha+\beta) + \alpha + (K-1)\cdot\beta, \tag{25}$$

where $S_{bc}(t)$ is the total transaction batch size at time slot t and L represents the average size of transactions.

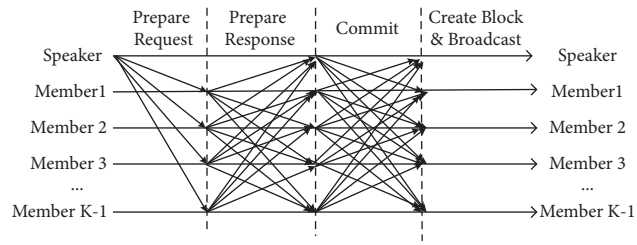

Figure 2: Consensus process of dBFT.

Then, if at least $K-f$ prepare response messages are received before the timeout, each consensus node verifies whether the messages are correct at first. Once the verification is successful, the commit messages are broadcast to other consensus nodes. During this phase, the consensus nodes verify $K-f$ signatures and MACs and then generate one signature and $K-1$ MACs for forming commit messages. Thus, for each consensus node, the consumed CPU cycles can be represented as

$$c_3(t) = (K-f)\cdot(\alpha+\beta) + \alpha + (K-1)\cdot\beta. \tag{26}$$

Finally, if the consensus nodes have collected more than $K-f$ commit messages and verified successfully, the consensus process is regarded as completed. At the same time, one block is produced and broadcast to blockchain system. During this phase, $K-f$ signatures and MACs should be verified by one consensus node. Thus, the computation cycles for each consensus node in this process are represented as

$$c_4(t) = (K-f)\cdot(\alpha+\beta). \tag{27}$$

In terms of above analysis, the delay of consensus process is represented by

$$T_{bc}(t) = \underbrace{\frac{c_1(t)}{f_{sp}(t)}}_{\text{Prepare Request}} + \underbrace{\max_{k\in K(k\neq sp)}\frac{c_2(t)}{f_k(t)}}_{\text{Prepare Request}} + \underbrace{\max_{k\in K}\frac{c_3(t)}{f_k(t)}}_{\text{Commit}} + \underbrace{\max_{k\in K}\frac{c_4(t)}{f_k(t)}}_{\text{CreatBlock\&Broadcast}} + T_i(t) + 3T_b(t), \tag{28}$$

where $f_{sp}(t)$ denotes the computing capacity of the speaker, $f_k(t)$ represents the computing capacity of the consensus node k, $T_i(t)$ is the block generation interval, and $T_b(t)$ is the broadcast delay between nodes.

3. Problem Formulation

In this section, it is necessary to jointly optimize routing path selection, offloading decision, and block size selection in a real-time environment so as to decrease delay and cost of the proposed network. The majorization issue is characterized as a MDP by identifying the state space, action space, and reward function.

3.1. State Space. During each time slot t, we define the state space as a union of the link quality between each pair of all trains $P_L(t) = \{P_{ij}(t)\}i, j = 1, 2, \ldots, V$, relaying price of all trains $\Pr(t) = \{\Pr_1(t), \Pr_2(t), \ldots, \Pr_V(t)\}$, computing resource of MEC servers $F(t) = \{F_1(t), F_2(t), \ldots, F_R(t)\}$, and computing price of MEC servers $Pc(t) = \{Pc_1(t), Pc_2(t), \ldots, Pc_R(t)\}$, which is represented as

$$S(t) = \{P_L(t), \Pr(t), F(t), Pc(t)\}. \tag{29}$$

3.2. Action Space. The action space involves the routing path selection, the offloading decision, and block size selection. Formally, the action space $A(t)$ is denoted as

$$A(t) = \{N(t), a_o(t), S_{bc}(t)\}, \tag{30}$$

where (t) is the set of relaying trains arranged in consecutive routing order in the multi-hop routing path. $a_0(t) = \{0, 1, \ldots, R-1\}$ indicates the offloading decision, and $a_0(t) = 0$ represents that the task is executed on the MEC server managed by the first supplier, while $a_0(t) = R-1$ indicates that the task is executed on the MEC server managed by the R-th supplier. $S_{bc}(t) \in \{1, 2, \ldots, S_{\max}\}$ represents different level for block size and $S_{\max}$ is the maximum block size.

3.3. Reward Function. We define the reward function to improve system performance and then devise immediate reward as

$$\begin{aligned} r(t) &= \begin{cases} w_1 \cdot \dfrac{1}{T_{sum}(t)} + w_2 \cdot \dfrac{1}{P_{sum}(t)}, \text{ if } C1 - C3 \text{ are satisfied,} \\ w_1 \cdot \dfrac{1}{T_{sum}(t)} + w_2 \cdot \dfrac{1}{P_{sum}(t)} - \vartheta, \text{otherwise,} \end{cases} \\ &s.t.\ C1:\ T_{sum}(t) \le \tau, \\ &C2:\ T_{bc}(t) \le \varepsilon \times T_i(t), \\ &C3:\ S_{bc}(t) \le S_{\max}, \end{aligned} \tag{31}$$

where w_1 and $w_2 \in [0, 1]$ are the weights of the latency and the cost, respectively, and $w_1 + w_2 = 1$. ϑ is the penalty value.

In this problem, $C1$ indicates the time limitation for completely offloading tasks, where τ is the maximum tolerable delay. $C2$ denotes the latency limitation of completing a block, where $\varepsilon > 1$. The maximum size of the all transactions in a single consensus process is indicated by $C3$.

4. Problem Solution

In this paper, due to high dynamic characteristics of the proposed system, we adopt the dueling DQN algorithm to solve the proposed joint optimization issue. Dueling DQN is widely considered as a significant improvement to conventional DQN. Different from the natural DQN, dueling DQN divides the Q-network into two parts, action advantage function with independent of state $A(s_t, a_t; \omega, \xi)$ and state-value function $V(s_t; \omega, \theta)$, which are calculated separately [20, 21]. It is easy to find which action has better feedback by learning $A(s_t, a_t; \omega, \xi)$. Finally, we can obtain the output of the dueling DQN network by merging two fully connected layers, which is denoted as

$$Q^{\pi}(s_t, a_t; \omega, \theta, \xi) = V^{\pi}(s_t; \omega, \theta) + A^{\pi}(s_t, a_t; \omega, \xi), \tag{32}$$

where ω is the convolution layer parameter, θ represents the parameter of the specific connected layer of the state-value function, and ξ denotes the parameter of the specific connected layer of the action advantage function. However, there is an unidentifiable problem in equation (32), which means that the respective roles of state-value function and action advantage function in the final Q value cannot be identified. To address that problem, dueling DQN sets expected value of the action advantage function to be zero at the selected action and implements the forward mapping of the last module of the network, which is written as

$$Q^{\pi}(s_t, a_t; \omega, \theta, \xi) = V^{\pi}(s_t; \omega, \theta) + A^{\pi}(s_t, a_t; \omega, \xi) - \frac{1}{|A|} \times \sum_{a'} A^{\pi}(s_t, a_t'; \omega, \xi). \tag{33}$$

The separation of environmental state value and action advantage in dueling DQN solves the problem of repeated calculation of the same state value, enhancing the capability of estimating the environmental state with a clear optimization objective [22]. Therefore, we adopt dueling DQN in our proposed network to decrease computational complexity and training time.

Finally, the training process is formally described in Algorithm 1.

(1) Initialization:
  Initialize the experience memory D and the mini-batch size B;
  Initialize evaluated network with the weight and bias set ω;
  Initialize target network with the weight and bias set ω^{-};
  Initialize the greedy coefficient ϵ;
(2) **for** $I = 1, \ldots, I_{\max}$ **do**
(3) Reset the state of trains and MEC servers with a random initial observation s_{ini}, and $s(t) = s_{ini}$;
(4) **for** $H = 1, \ldots, H_{\max}$ **do**
(5) Randomly choose a probability p;
(6) **if** $p < \epsilon$ **then**
(7) Randomly choose an action $a(t) \neq a^*(t)$ based on ϵ-greedy policy;
(8) **else**
(9) $a(t) = a^*(t) = \text{argmax}_{a \in A} Q(s', a'; \omega', \theta', \xi)$;
(10) **end if**
(11) Execute action $a(t)$ and obtain the reward $r(t)$, and proceed to the next observation $s(t+1)$;
(12) Store the experience $(s(t), a(t), r(t), s(t+1))$ into experience replay memory;
(13) Randomly sample a mini-batch of $(s(i), a(i), r(i), s(i+1))$ from experience replay memory D;
(14) Obtain two parts of evaluated network, including $V(s(t))$ and $A(a(t))$, and merge them as $Q(s(t), a(t); \omega, \theta, \xi)$ through equation (33);
(15) Obtain target Q value in target network by $Q_{\text{target}}(s) = re(t) + \gamma \max_{a \in A} Q(s', a'; \omega', \theta', \xi)$;
(16) Train evaluated network to minimize loss function $L(\omega)$ by $L(\omega) = E[(Q_{\text{target}}(s) - Q(s', a'; \omega', \theta', \xi'))^2]$;
(17) Every several training steps, modify target Q-network according to evaluated Q-network;
(18) $s(t) \longrightarrow s(t+1)$;
(19) **end for**
(20) **end for**

Algorithm 1: Performance optimization framework for MEC-enabled smart rail system by multi-hop data transmission and blockchain based on dueling DQN.

5. Simulation Results and Discussion

In this part, we depict the effectiveness of the proposed scheme through simulation experiments. Firstly, the simulation environment and parameters are presented. Then, we analyze and discuss the results and the performance of the proposed framework.

5.1. Simulation Parameters. In the simulation experiments, we consider the network scenario with five trains running on the track, as well as two MEC servers managed by different suppliers. Furthermore, we summarize other significant simulation parameters in Table 1.

In order to assess how well the proposed framework performs, we consider five comparison schemes as follows. (1) The routing path is picked at random in the proposed method without path selection. (2) An approach without offloading selection: the MEC servers conduct the computing tasks at random. (3) Block sizes for created blocks are fixed in the proposed approach. (4) A technique based on natural DQN solves the problem as it is formulated. (5) PBFT-based scheme: all blockchain nodes participate in the consensus process.

5.2. Performance Comparison of Convergence. The convergence of the proposed optimization framework under various learning rates is shown in Figure 3. As can be seen in this figure, the learning rate (10^{-1}) performs better than other schemes. It is because the large learning rate (10) might fall into local optimum and fail to obtain the globally optimal solution of the proposed problem. Moreover, the small learning rate (10^{-7}) likely led to the slow convergence rate and took longer to find the optimal value. Hence, in this paper, the learning rate is selected carefully and set to be 10^{-1}.

As shown in Figure 4, we examine the convergence performance under the different algorithms. It can be observed that dueling DQN reaches higher system reward and performs more stably than the scheme with natural DQN. The reason is that the chosen routing path, the selected MEC server, and the selected block size can hardly affect the changes of state in our scenario. Thus, our proposed scheme with dueling DQN has more advantage to the decision of the agent in this case.

Figure 5 depicts the comparison of system reward with training steps under our proposed dBFT-based scheme and PBFT-based scheme. We can see that dBFT-based scheme gets higher total reward. It is because all nodes need to participate in the consensus process under the PBFT-based scheme. On the contrary, there is only a part of the trusted nodes participating for consensus under the dBFT-based scheme. The dBFT algorithm reduces the computation cycles and improves the efficiency of the consensus process. Therefore, our proposed dBFT-based scheme is more suitable for the smart rail system with high-speed mobility.

5.3. Performance Comparison of Different Aspects. Figure 6 presents the relationship between total latency and task data size under different schemes. One observation is

Table 1: Experiment parameters.

Parameters	Value
The number of nodes V	5
CPU cycles required to generate and verify a signature α	2 Mcycles [23]
CPU cycles required to generate and verify a MAC β	1 Mcycles [23]
Average transaction size L	20 kB
Average computing resources of MEC servers $F_R(t)$	4×10^{10}–6×10^{10} CPU cycles/s [2]
Size of offloading task $B_{\mathrm{MV}}(t)$	500 kB
Computation density of application	200 CPU cycles/bit [2]
Unit price for data relaying $\Pr(t)$	500–1000
Unit price for data computation $Pc(t)$	2000–3000
Weights of latency w_1	0.6
Weights of cost w_2	0.4
Learning rate	10^{-1}

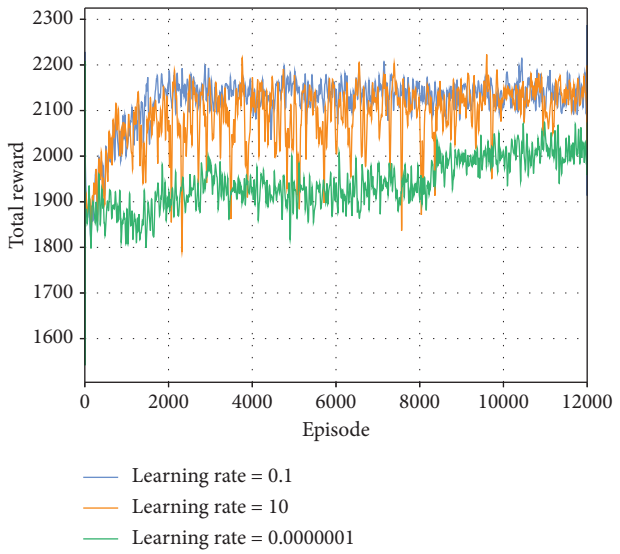

Figure 3: Total reward with various learning rates.

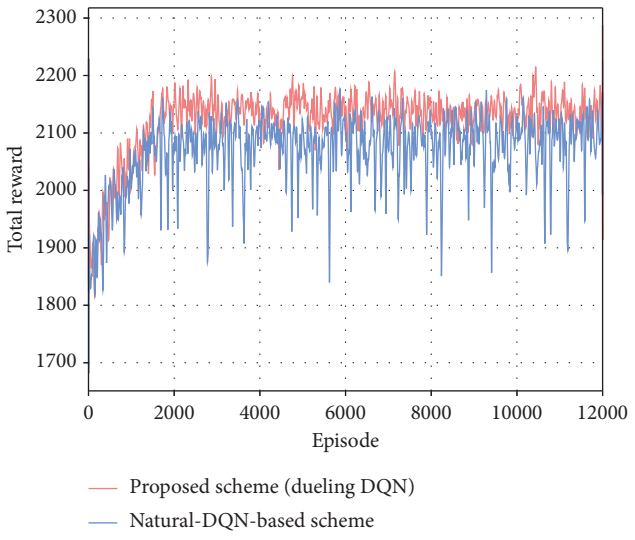

Figure 4: Total reward with diverse learning algorithms.

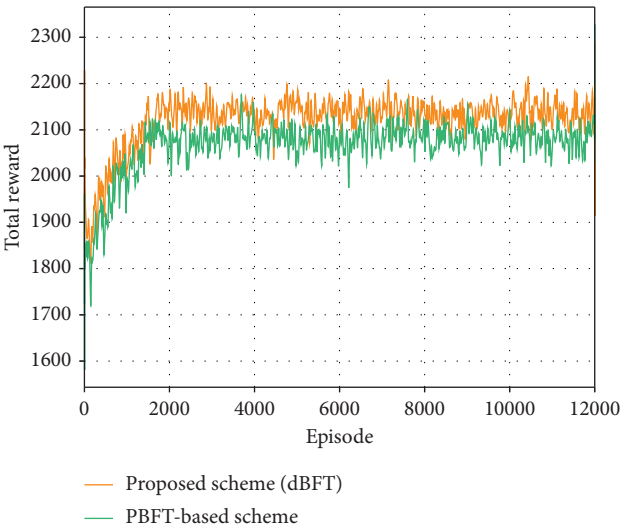

Figure 5: Total reward under different consensus mechanisms.

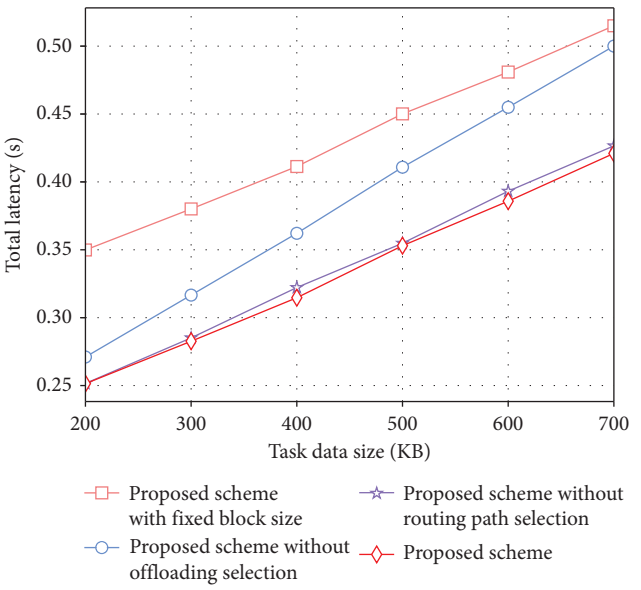

Figure 6: Total latency versus task data size under various schemes.

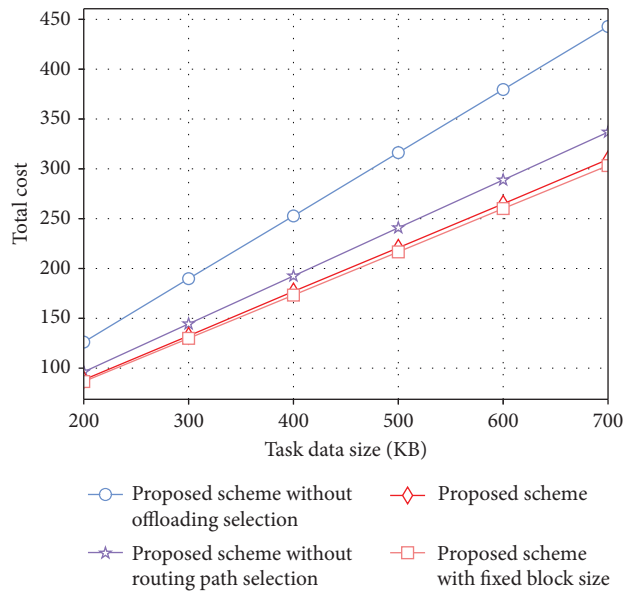

Figure 7: Total cost versus task data size under various schemes.

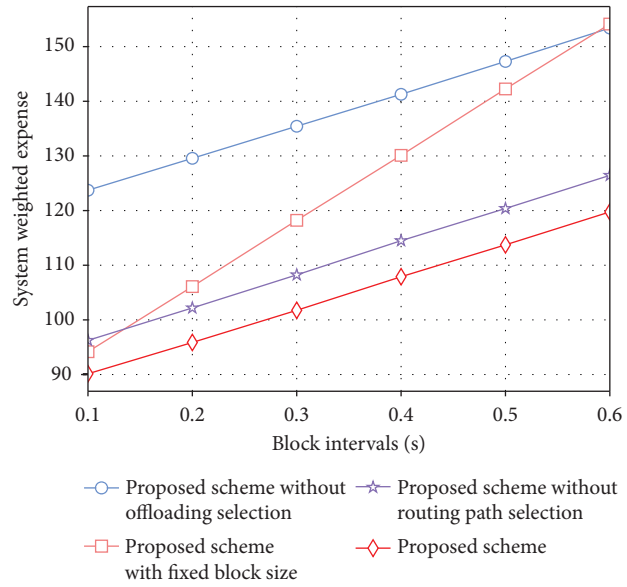

Figure 8: System weighted expense versus block intervals under various schemes.

that total latency under all schemes increases with the increase of task data size. The reason is that the increase in task size takes longer for end-to-end delivery and offloading computation. Moreover, we can see from this figure that the total latency under our proposed scheme is consistently lower than that of others. It is because our proposed scheme simultaneously optimizes the routing path selection, offloading strategy, and block size selection. On the contrary, previous baselines just optimize a portion of these items.

Figure 7 illustrates the comparison of total cost with the size of the task data under various schemes. We can see that as the task data size increases, so does total cost of all schemes. Furthermore, our proposed scheme is superior to

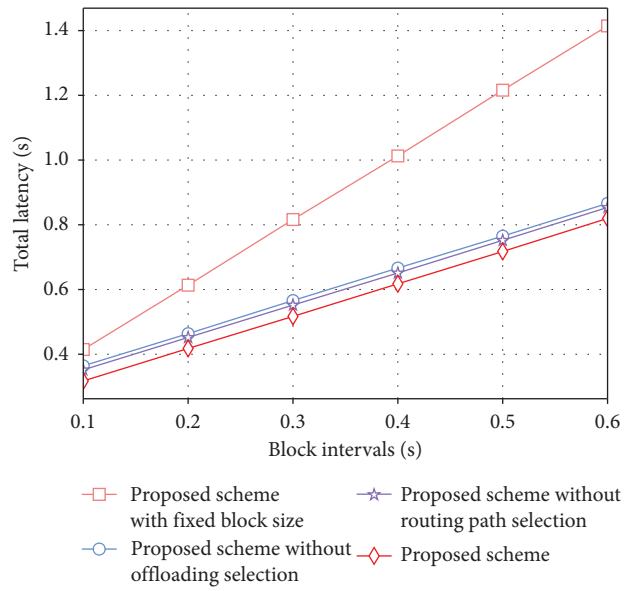

Figure 9: Total latency versus block intervals under various schemes.

the schemes without routing path selection and offloading selection. Nevertheless, the scheme without fixed block size performs better than the proposed scheme. The reason is that the state of link quality fluctuates according to the high-speed movement of trains.

As shown in Figures 8 and 9, we examine the system weighted expense and the total latency under different block intervals. It can be observed that all the schemes gain a higher system weighted expense and total latency with the increase of the block intervals. The overall system latency and cost make up the system weighted expense. It is because the block generation interval rises, which makes the delay of blockchain higher. Additionally, system weighted expense and total latency of the previous baselines are visibly higher than those of our proposed scheme. Therefore, with joint consideration of the adaptive routing path selection, the optimal offloading decision, and the appropriate block size selection, our proposed scheme acts the best compared with other schemes.

6. Conclusions

In this paper, an improved optimization framework for the MEC-enabled smart rail system is proposed. In order to enable the MEC servers to be utilized in a wider range while meeting the low-latency requirement, multi-hop ad hoc network was applied to our proposed network model. Moreover, the blockchain technology based on the dBFT consensus mechanism was considered and introduced to effectively guarantee the security and reliability during multi-hop data transmission. Then, in order to reduce system latency and cost, the routing path selection, offloading strategy, and block size selection are co-optimized. We described the proposed dynamic majorization problem as a MDP and adopted dueling DQN to solve it. Simulation results demonstrated that the performance of the proposed

scheme is better than existing baseline schemes. Furthermore, in the future, other routing mechanisms and cloud-edge collaborative architecture would be considered in our multi-hop ad hoc network for the smart rail system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported in part by the National Natural Science Foundation of China under grant no. 61901011, in part by the Beijing Natural Science Foundation under grant nos. L211002, 4222002, and L202016, and in part by the Foundation of Beijing Municipal Commission of Education under grant nos. KM202110005021 and KM202010005017.

References

[1] H. Tu, "Research on the application of cloud computing technology in urban rail transit," in *Proceedings of the 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, pp. 828–831, Dalian, China, August 2020.

[2] Y. X. Liu, *Research on End-Edge-Cloud Collaborative Computing for Intelligent Railway*, Beijing Jiaotong University, Beijing, China, 2021.

[3] K. Zhang, Y. M. Mao, S. P. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: a promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[4] Y. Li, L. Zhu, H. Wang, F. R. Yu, and S. Liu, "A cross-layer defense scheme for edge intelligence-enabled CBTC systems against MitM attacks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2286–2298, 2021.

[5] H. C. Zhang, *Research on Computing Offloading Method Based on Mobile Edge Computing in Internet of Vehicles*, Zhejiang University of Science and Technology, Hangzhou, Zhejiang, China, 2020.

[6] J. Balen, G. Martinovic, K. Paridel, and Y. Berbers, "PVCM: assisting multi-hop communication in vehicular networks using parked trains," in *Proceedings of the 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems*, pp. 119–122, St. Petersburg, Russia, October 2012.

[7] L. Zhu, H. Liang, H. Wang, B. Ning, and T. Tang, "Joint security and train control design in blockchain-empowered CBTC system," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8119–8129, 2022.

[8] J. F. Xie, H. Tang, T. Huang et al., "A survey of blockchain technology applied to smart cities: research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2794–2830, 2019.

[9] J. Kang, R. Yu, X. Huang et al., "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, 2019.

[10] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the internet of things: research issues and challenges," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2188–2204, 2019.

[11] L. Zhu, Y. Li, F. R. Yu, B. Ning, T. Tang, and X. Wang, "Cross-layer defense methods for jamming-resistant CBTC systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, pp. 7266–7278, 2021.

[12] X. M. Zhang, K. H. Chen, X. L. Cao, and D. K. Sung, "A street-centric routing protocol based on microtopology in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 7, pp. 5680–5694, 2016.

[13] X. M. Zhang, X. L. Cao, L. Yan, and D. K. Sung, "A street-centric opportunistic routing protocol based on link correlation for urban VANETs," *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1586–1599, 2016.

[14] Z. Z. Deng, Z. Cai, and M. G. Liang, "A multi-hop VANETs-assisted offloading strategy in vehicular mobile edge computing," *IEEE Access*, vol. 8, pp. 53062–53071, 2020.

[15] A. Boualouache, H. Sedjelmaci, and T. Engel, "Consortium blockchain for cooperative location privacy preservation in 5G-enabled vehicular fog computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 7, pp. 7087–7102, 2021.

[16] Z. H. Xue, *Improvement and Research of Blockchain Consensus Algorithm Based on Byzantine Fault Tolerant*, Lanzhou Jiaotong University, Lanzhou, Gansu, China, 2021.

[17] G. J. Han, J. F. Jiang, L. Shu, and M. Guizani, "An attack-resistant trust model based on multidimensional trust metrics in underwater acoustic sensor network," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2447–2459, 2015.

[18] J. W. Kang, Z. H. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao, "Toward secure blockchain-enabled internet of vehicles: optimizing consensus management using reputation and contract theory," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2906–2920, 2019.

[19] J. Feng, Q. Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1320–1323, 2019.

[20] Z. H. Xuan, G. Y. Wei, and Z. W. Ni, "Power allocation in multi-agent networks via dueling DQN approach," in *Proceedings of the 2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)*, pp. 959–963, Nanjing, China, October 2021.

[21] W. L. Jiang, C. Bao, G. Q. Xu, and Y. Wang, "Research on Autonomous Obstacle Avoidance and Target Tracking of UAV Based on Improved Dueling DQN Algorithm," in *Proceedings of the 2021 China Automation Congress (CAC)*, pp. 5110–5115, Beijing, China, October 2021.

[22] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu, and C. Zhao, "Blockchain-based software-defined industrial internet of things: a dueling deep ${Q}$ -learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4627–4639, 2019.

[23] F. X. Guo, F. R. Yu, H. L. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1689–1703, 2020.