A Multi-Agent Reinforcement Learning-Based Task-Offloading Strategy in a Blockchain-Enabled Edge Computing Network

Chenlei Liu \(^1,2,3,†\) and Zhixin Sun \(^1,2,3,*,†\)

1. Introduction

Mobile Edge Computing (MEC) is an emerging computing paradigm that deploys computing resources at the edge of the network to provide low-latency, high-bandwidth, and customizable services that can deliver computing and storage capabilities at the edge of the network, close to the data source. In addition, MEC can improve the performance, efficiency, and security of various applications that require low latency, high bandwidth, and data privacy, such as augmented reality, smart cities, and autonomous driving. However, because the mobile edge computing network environment has the characteristics of openness and dynamics, the network is vulnerable to the threat of malicious node invasion and data attacks. Such attacks can cause problems such as shared data leakage, task execution interference, and resource allocation anomalies within the network, seriously...
affecting the security of MEC. Therefore, ensuring the safe sharing of data and trustworthy collaboration of nodes is an important issue that needs to be solved in MEC [1,2].

Blockchain, known as a distributed, tamper-proof, decentralized data storage technology, was first proposed by Nakamoto in the context of Bitcoin, which enables secure, transparent, and immutable transactions and transfer of data records between multiple parties without relying on a trusted third party [3]. Blockchain has been widely used in various fields, such as digital asset management, supply chain finance, and intelligent manufacturing. Blockchain-based MEC (BMEC) is a new type of architecture that applies blockchain technology to MEC systems and can solve many challenges, such as data security, privacy protection, incentive mechanisms, resource management, etc. [4]. The in-depth integration of blockchain and MEC has been widely discussed [5]. In telematics and intelligent transportation systems, blockchain can provide collaborative management of service resources [6], data security sharing management [7], and collaborative node identity authentication [8]. In the smart grid, blockchain-based MEC is mainly applied to system architecture design [9], energy transaction pricing [10], and transaction security [11]. In addition, benefiting from the advantages of blockchain-based MEC, intelligent health care [12] and artificial intelligence [13] are also beginning to be applied.

Although the blockchain-based MEC system has excellent application prospects and research value, it also faces some important problems, including task offloading. MEC task offloading refers to the technology of offloading computing tasks from user devices to edge nodes or clouds for execution in order to solve the deficiencies of user devices in terms of resource storage, computational performance, and energy efficiency. For the problem of task offloading in blockchain-based MEC systems, there are still some limitations in the current research, mainly with respect to the following aspects: (1) most existing works only consider the quality of service indicators of traditional mobile edge computing task offloading, such as task processing latency and energy consumption, but ignore blockchain mechanisms and user privacy leakage, which makes the problem modeling insufficient [14–16], and (2) task offloading algorithms are often based on heuristic learning methods or single-agent reinforcement learning algorithms [17–19]. The analytical performance and solution efficiency need to be more satisfactory for dynamically changing, high-dimensional, non-convex task-offloading problems. In this paper, blockchain-based mobile edge computing task-offloading modeling and a multi-agent reinforcement learning method are investigated, and the main innovative contributions are summarized as follows:

- We propose a novel task-offloading model for blockchain-based MEC networks that comprehensively considers the blockchain-specific incentive mechanism and consensus mechanism. It also takes the user privacy metric as the optimization objective, together with the task service quality as the joint optimization objective, which makes the modeling of the optimization problem more in line with the practical environment;
- We propose a reinforcement learning algorithm based on a multi-agent global memory pool. Agents can enhance the overall collaborative ability among the agents by sharing parameters;
- We adopt attenuatable Gaussian random noise in the action space selection process in the actor network to enhance the search capability and avoid falling into local optimum;
- We conduct several sets of comparative experiments to validate the performance of the proposed algorithm in dealing with the task-offloading problem.

This paper is structured as follows. Section 2 investigates state-of-the-art research related to the research content of this paper. Section 3 presents the proposed blockchain MEC network task-offloading optimization model to be solved in this paper. Section 4 describes the principle and process of the reinforcement learning algorithm used in this paper. Section 5 conducts simulation experiments to evaluate the performance and effectiveness of the proposed algorithm. Section 6 summarizes the full paper.
2. Related Works

Blockchain-based MEC networks constitute an emerging research field combining the decentralization, high security, and tamper-proof features of blockchain and the low-latency, high-bandwidth, and real-time advantages of MEC to provide ideas for solving a series of challenges faced by the MEC network structure, such as security, privacy protection, resource management, and so on. This combination is crucial for building the next-generation of intelligent, secure, and efficient network environments. It has attracted many scholars to research its architecture and operation mechanism in depth. Le et al. [20] established a unified six-layer architecture with high efficiency, security, compatibility, and flexibility for blockchain-based resource sharing and transactions in mobile access networks. The proposes architecture contains many new features, such as an enhanced blockchain structure, secure interaction methods, efficient service mechanisms, and scalable transaction models. Salim et al. [21] proposed a latency tolerance-based cybertwin-assisted task-scheduling scheme, where cybertwins using logger functionality and digital asset functionality exchange smart contracts with cloud operators using digital assets to ensure maximum computational resources for efficient job allocation to edge clouds. Sun et al. [22] considered incentive and cross-server resource allocation in blockchain-driven MEC, where the blockchain prevents malicious edge servers from tampering with player information by maintaining a continuous, tamper-proof ledger database. In addition, they proposed two double auction mechanisms, namely the break even-based double auction mechanism (DAMB) and the more efficient break-even free double auction mechanism (BFDA), in which users request multitasking services with declared bids and edge servers cooperate to serve the users. Ding et al. [23] proposed a new noma-based MEC wireless blockchain network that minimizes system energy consumption through task offloading decision optimization, user clustering, computational resources, and transmission power allocation. Zhang et al. [24] proposed a reliable and efficient system based on edge computing and blockchain and designed a new group agent strategy based on trust computing that ensures the edge devices in the process of interaction to ensure reliability and improve transmission efficiency.

Task offloading is an important computing strategy that allows mobile devices to shift computationally intensive tasks to be executed on more powerful remote servers or edge computing nodes to reduce the computational burden and energy pressure on mobile devices while increasing task processing speed and efficiency, which is particularly important in MEC environments. This is because MEC enables low-latency, high-bandwidth services by bringing computing resources and storage capacity to the edge of the network, i.e., close to users and data sources. However, some challenges are still associated with introducing blockchain technology into MEC networks. The resource allocation process of MEC involves parameters such as latency, resource utilization, service provider profit, service user satisfaction, and energy. Blockchain parameters such as throughput, block size, block time, and block reward also need to be considered while designing blockchain-based resource allocation systems. Blockchain ensures decentralization, transparency, and invariance but also introduces computational and communication overheads and increases latency. Therefore, joint optimization of MEC resource allocation and blockchain parameters is an open challenge [25]. In addition, data sharing and knowledge discovery are essential requirements for the integration of blockchain and edge computing systems in many application scenarios, e.g., the need to comprehensively analyze a large number of patient medical records in smart health care and the need for vehicles to share GPS positioning data in the Internet of Vehicles (IoV) in order to correct errors in assisted autonomous driving. However, as the number of devices increases and device-generated data become more decentralized, an explosive increase in network nodes drives a huge demand for data sharing due to the large amount of sensitive information and private data stored in the system. Therefore, protecting data privacy and security is an important challenge for blockchain-based MEC [26]. Therefore, exploring the effective combination of blockchain technology and MEC for a secure, efficient, and scalable task-offloading mechanism re-
mains an active and challenging research area [27]. For example, Guo et al. [28] used the Stackelberg game to model the interactions between edge cloud operators and different collaborative mining networks to obtain the optimal resource prices and device resource requirements when offloading tasks to the edge cloud. Lin et al. [14] proposed an efficient Device-to-Device (D2D) network authorization blockchain framework and designed an elastic resource allocation scheme using Lyapunov optimization theory to achieve high throughput with limited resources. Zhang et al. [15] proposed an efficient and improved closed-ended quadratic bidding game for allocating communication and computation resources under the quality-of-service (QoS) constraints of smart terminals in response to the optimization problem of joint communication and computation resource allocation, thus creating an edge cloud resource-sharing model based on blockchain technology and an auction game. Devi et al. [29] developed a system model to solve the task-offloading algorithm to minimize the data center’s completion time and energy consumption. In addition, blockchain-based, energy-aware task scheduling for data centers was proposed to provide the best solution for minimizing completion time and energy consumption. These works have investigated blockchain-based task offloading for MEC at the level of heuristic algorithms and mathematical computational methods such as game theory. However, these schemes have also become increasingly difficult to apply due to the random mobility of user terminals in the edge computing network, task uncertainty, and the complexity of the optimization problem, requiring the consideration of task offloading with multiple optimization objectives.

In recent years, due to the random mobility of user terminals in edge computing networks, the uncertainty of tasks, and the complexity of optimization problems, the task-offloading problem, which needs to consider multiple optimization objectives, has become increasingly challenging to be solved by traditional heuristics and game-theoretic methods. Deep reinforcement learning-based methods have gradually been widely applied to research with the aim of solving the traditional edge computing network task-offloading problem [30–33] and gradually extended and applied to research on task offloading for blockchain-based MEC networks. Yang et al. [34] integrated MEC (MEC) into a blockchain-based industrial IoT system to improve the computational power of industrial IoT devices with a comprehensive consideration of weighted system cost, including energy consumption and computational overheads, and formulated the posed problem as a Markov Decision Process (MDP), introducing Deep Reinforcement Learning (DRL) to solve the formal problem. Although the study demonstrated the effectiveness of the method in small-scale networks, further validation is needed for scalability and practical deployment in large-scale networks. Nguyen et al. [35] proposed a new distributed deep reinforcement learning-based approach employing a multi-agent deep deterministic policy gradient algorithm. Based on this, a game-theoretic solution was built to model the offloading and mining competition between edge devices as a potential game and prove the existence of a pure Nash equilibrium. However, the experimental scenarios and parameter settings in the study may not fully reflect the complexity of practical applications, and further verification of its generalization ability in different environments is needed. Yao et al. [36] proposed a blockchain-empowered collaborative task-offloading scheme for cloud-edge-device (CE-device) computation by modifying the blockchain consensus process to enable participants to reach a formulaic agreement by solving the task-offloading problem. To this end, each participant can apply a reinforcement learning-based approach to solve the task-offloading problem and compete for the block output right by comparing the performance of the offloading strategies and accepting the best strategy as the offloading solution for the next period. However, the scenario does not discuss the impact of operations in the blockchain on the task-offloading strategy. Nguyen et al. [19] proposed a reinforcement learning-based multi-user task-offloading algorithm to obtain a dynamic blockchain network with MEC using the optimal offloading algorithm. The scheme formulates task offloading and privacy protection as a joint optimization problem and employs a reinforcement learning-based Q-network algorithm to learn the offloading strategy, which minimizes the total system cost.
in terms of combined computational latency and energy consumption while guaranteeing optimal user privacy and mining reward performance. However, the study adopted a single-agent reinforcement learning scheme, which is questionable for the computational efficiency of task offloading in complex network environments. Wang et al. [37] proposed a deep reinforcement learning (DRL)-based support scheme for blockchain-based IoT resource orchestration in which IoT edge servers and end users can reach a consensus on network resource allocation based on blockchain theory. In addition, agents relying on a policy network can be trained with these resource attributes to fully perceive changes in the network state and, thus, make dynamic resource allocation decisions. However, the study only considered the performance metrics of mobile edge computation tasks under a single user, and the user’s mobile characteristics need to be fully considered. In summary, these works enrich the research on blockchain–MEC task offloading based on reinforcement learning but still have certain defects that need to be improved in subsequent research.

This paper investigates a blockchain-based task-offloading model for MEC networks that takes the incentive mechanism and consensus mechanism of blockchain into account in the task-offloading problem model and, at the same time, takes the user privacy metric and the task service quality as the joint optimization objectives so that the optimization problem is more in line with the practical environment. In addition, an actor–critic reinforcement learning algorithm based on a shared global memory pool of multiple agents is proposed to improve the robustness and stability of the performance by sharing parameters. An action space selection process based on Gaussian noise is added to increase the algorithm’s spatial search ability to avoid falling into a local optimum.

3. Model

In this section, we propose a blockchain-based MEC system architecture, then provide a system overview and describe the operational flow and the blockchain consensus process.

3.1. System Model

In this paper, we propose a blockchain mobile edge network task-offloading model. The specific architecture of this model is shown in Figure 1. The network model mainly contains a blockchain layer, an edge server layer, and a device layer. The device layer contains a collection of devices, and different user devices interact with the MEC network environment, send task offload requests to the edge service node, and receive the offload policy feedback from the edge service node to complete task offloading. The edge server layer contains a collection of edge nodes; each node has certain task processing resources, receiving task-offloading requests sent by users and completing the task-offloading requests of the devices through cooperation between nodes. The edge service nodes also have the role of blockchain nodes, which can participate in network consensus and reward allocation in the blockchain layer and jointly maintain the blockchain that stores network information, ensuring the security of the network and incentivizing the participation of nodes.

The edge network of this model has a device set \( U = \{ u_1, u_2, \ldots, u_n \} \) consisting of \( n \) user devices and an edge node set \( E = \{ e_1, e_2, \ldots, e_m \} \) consisting of \( m \) edge nodes. For any user device, \( u_i = (pw_{li}, f_{li}, \text{enc}_{i}^{\text{max}}, tk_{li}, \text{loc}_{i} = (x_{li}, y_{li})) \) can move along the irregular trajectory within the time slot and initiate a task-offloading request to the edge server, where \( pw_{li} \) is the total transmission power of the device, \( f_{li} \) is the processing speed (the number of processing cycles per second), \( \text{enc}_{i}^{\text{max}} \) is the upper energy limit of the device, \( tk_{li} \) is the task initiated by the user in time slot \( t \), \( \text{loc}_{i} \) is the device localization, and \( x_{li} \) and \( y_{li} \) are the position coordinates.

In addition, the task \( (tk_{li}) \) of any user device can be expressed as an array \( ((d_{li}^t, D_{li}^t, Td_{i,j}^{\text{max}}) \), where \( d_{li}^t \) (bit) is the size of the task, \( D_{li}^t \) is the number of computation cycles required for the computation task (500 CPU computation cycles are required to process the data of a 1 bit task in this paper), and \( Td_{i,j}^{\text{max}} \) is the maximum tolerable delay of the task). Due to the limitation of energy consumption and computational capability of the device, these tasks cannot all be computed locally at the same time and need to be partially offloaded to the
edge node by using $tk_{ij}$ and $tk_{ij}^{o}$ to denote the local computation of the task ($tk_{ij}$) and its offloading part to the node ($e_j$), respectively. The sizes of the corresponding offloading task and computational task cycle are denoted as $d_{ij}^{l}$, $D_{ij}^{l}$, $d_{ij}^{o}$, and $D_{ij}^{o}$, respectively.

For any edge node, $e_j = (k_j, pw_{ej}, f_{ej}, loc_{ej} = (x_{ej}, y_{ej}))$ can receive the task data offloaded by the device and process the task using its computational resources, then return the result to the smart device after task processing is completed. $k_j$ denotes the number of tokens held by the block node corresponding to the edge node, $pw_{ej}$ denotes the transmission power of the edge node, $f_{ej}$ denotes the processing speed of the edge node, $loc_{ej}$ denotes the fixed location of the server, and $x_{ej}$ and $y_{ej}$ are the location coordinates.

In the blockchain of this model, all the mobile edge network nodes also have the role of blockchain nodes, sharing parameters and recording proof of workload through the blockchain. The consortium blockchain uses a Proof of Stake (PoS)-based consensus method to validate the workload of the computing nodes and distribute incentive rewards to each of the individual nodes involved in the computation of the offloading task.

### 3.2. Consensus Model

The consensus mechanism in blockchain is the core method to ensure that all participating nodes agree on the state of the blockchain. Currently, there are two main consensus mechanisms in mainstream blockchain systems, namely Proof of Work (PoW) and proof of stake (PoS). In PoW, all entities compete to solve a mathematical puzzle to generate blocks and receive a reward. However, the process of PoW is very computationally intensive and only applies to mobile edge network scenarios. PoS is proposed to address the limitations of PoW, and unlike PoW, the probability of an entity getting the right to publish a block depends on its equity, i.e., the number of tokens owned by the entity [38]. A comparison of the two consensus mechanisms is shown in Table 1.

---

**Figure 1.** Blockchain-based edge computing network model.

---

---
Table 1. Consensus mechanism comparison.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Proof of Work (PoW)</th>
<th>Proof of Stake (PoS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy consumption</td>
<td>High; requires large amounts of power</td>
<td>Low; does not require large amounts of computing resources</td>
</tr>
<tr>
<td>Hardware requirements</td>
<td>Requires high-performance hardware</td>
<td>No high-performance hardware required</td>
</tr>
<tr>
<td>Attack cost</td>
<td>High; needs to control 50% of computing power of the whole network</td>
<td>High; needs to control 50% of the tokens of the whole network</td>
</tr>
<tr>
<td>Reward mechanism</td>
<td>Mining rewards (blockchain currency)</td>
<td>Token rewards</td>
</tr>
<tr>
<td>Block generation speed</td>
<td>Usually slow; affected by computational difficulty</td>
<td>Usually slow; affected by computational difficulty</td>
</tr>
<tr>
<td>Degree of decentralization</td>
<td>High but tends to be concentrated in mining pools</td>
<td>High; coin holders are more widely distributed</td>
</tr>
</tbody>
</table>

In this paper, the PoS-based consensus mechanism is used to implement the workload consensus checking of computing nodes. Its execution process is as follows:

1. Packing node selection: The system selects a negative blockchain node \( e_2 \in E \) to construct a new block by periodically selecting the block creation node \( e_2 \in E \) among all the nodes with equity based on the number of tokens held by the verifier;

2. New block creation: The block creation node packages all blockchain network transactions in the system during time slot \( t \) into a new block, assuming that the block consists of the following two parts: the task \( (tk^i) \) offloading data \( (d^t_i) \) and the block-fixing data \( (d_0) \) contained in block \( b_i \). The size of the task transaction data is calculated from the original size conversion of the task, noting the conversion rate as \( s \). Then, the block size \( d^t_{b_i} \) can be expressed as

\[
d^t_{b_i} = d^t_i + d_0 = s \sum_{e_j \in E} d^{t_0}_{ij} + d_0
\]  

3. Block validation: The coalition chain calculates the selection probability of the edge nodes according to the number of tokens owned by the nodes using a Poisson distribution with parameter \( \lambda \). The first \( v \) nodes according to the order of probability constitute the set of validation nodes \( (E^V) \), in which the probability distribution of the edge node \( (e_j) \) being selected as a validation node is

\[
p^v_{ij} = P(K = k_j) = \frac{\lambda^k}{k!} e^{-\lambda}
\]

4. Block addition: Once a new block is recognized by all the validation nodes, it is added to the blockchain;

5. Incentive distribution: Based on the incentive mechanism, a certain reward is provided to the network nodes that participate in the task to compute and verify the new block.

The workflow of the blockchain-based MEC task-offloading system described in this section is shown in Figure 2.

3.3. Quality of Service Model

In this section, the blockchain-based MEC task-offloading quality of service model proposed in this paper is described in detail, in addition to description of the design methodology for quality of service models reported in existing MEC task-offloading research \([15,29,33]\), to simulate a blockchain–mobile edge network within each time slot. The delay and energy consumption generated by user task computation, task offloading communication, block verification, etc., are investigated to construct a blockchain-based network quality of service-oriented communication model and a computation model.
In this paper, it is assumed that the size of the task calculation result is much smaller than the task itself and that the communication overhead required to transmit the result is negligible. Therefore, this paper mainly considers the two data communication scenarios of user device task offloading and block verification and calculates the energy consumption and transmission delay in the communication process.

In this system model, the device and the MEC server are linked through a wireless network, and the transmission rate between them is affected by the transmission environment, communication resources, and transmission distance. In this paper, we refer to [40,41] and calculate the channel gain \( (h_{t_{ij}}) \) from any device \( u_i \) to edge node \( e_j \) in time slot \( t \) using the following formula:

\[
h_{t_{ij}} = \frac{h_0}{dist_{t_{ij}}^2}
\]  

(3)

where \( h_0 \) denotes the initial gain of the channel, \( \varphi \) is the path loss exponent, and \( dist_{t_{ij}} = \sqrt{(x_{i_{t}} - x_{e})^2 + (y_{i_{t}} - y_{e})^2} \) denotes the distance from device \( u_i \) to edge node \( e_j \) at time slot \( t \).

The signal-to-interference-plus-noise ratio (\( SINR_{t_{ij}} \)) from device \( u_i \) to edge node \( e_j \) is

\[
SINR_{t_{ij}} = \frac{pw_{t_{ij}} h_{t_{ij}}^2}{\sum_{e_j' \in E \setminus \{e_j\}} pw_{t_{ij'}} h_{t_{ij'}}^2 + N_0}
\]

(4)

where \( pw_{t_{ij}} \), \( N_0 \), and \( B \) denote the transmission power from device \( u_i \) to edge node \( e_j \) in time slot \( t \), the Gaussian noise in the channel, and the channel communication bandwidth, respectively. \( pw_{t_{ij}} = \sum_{e_j' \in E} pw_{t_{ij'}} \), and the data transmission rate from device \( u_i \) to edge node \( e_j \) in time slot \( t \) is

\[
R_{t_{ij}} = B \cdot \log_2(1 + SINR_{t_{ij}})
\]

(5)

Therefore, in the task-offloading communication scenario, the communication delay (\( Td_{t_{ij,comm}}^{t_{ij}} \)) and energy consumption (\( En_{t_{ij,comm}}^{t_{ij}} \)) of the user device \( (u_i) \) transmitting the task offloading to the edge node \( (e_j) \) during the time slot \( t \) is expressed as follows:

\[
Td_{t_{ij,comm}}^{t_{ij}} = \frac{d_{t_{ij}}^{t_{ij}}}{R_{t_{ij}}}
\]

(6)
During the block consensus process, the block generation node transmits the block to the validation node for verification. Assuming that the block generation node (e.g.) and validation nodes have a fixed network transmission speed (R) between them, the consensus verification communication delay ($T_{d_{ij,\text{comm}}}^{v}$) and energy consumption ($E_{t_{ij,\text{comm}}}^{v}$) between the block generation node (e.g.) and the validation node (e.v) for block $b$ are

$$T_{d_{ij,\text{comm}}}^{v} = \frac{d_{hi}}{R}$$

$$E_{t_{ij,\text{comm}}}^{v} = pw_{eg}vTd_{ij,\text{comm}}^{v} = pw_{eg} \frac{d_{hi}}{R}$$

3.3.2. Computing Model

In this paper, we mainly consider three kinds of computing scenarios, namely local task computing, task-offloading computing, and block verification computing. The computing model must determine the processing delay and energy consumption according to the computing process. It is assumed that the blockchain selects block generation nodes according to the number of tokens owned by the nodes, and the calculation volume of generation node selection is ignored in this model.

In the task computation scenario locally executed by the user device, the energy consumption coefficient of the user device is assumed to be $\varepsilon_l = 10^{-1[42]}$ in this paper. The delay ($T_{d_{ij,\text{comp}}}^{l}$) and energy consumption ($E_{t_{ij,\text{comp}}}^{l}$) of device $u_i$ for local task processing are

$$T_{d_{ij,\text{comp}}}^{l} = \frac{D_{ij}^{l}}{f_{li}}$$

$$E_{t_{ij,\text{comp}}}^{l} = \varepsilon_l D_{ij}^{l} (f_{li})^{2}$$

In the offloading task scenario executed by edge nodes, this paper assumes that the edge node provides a separate CPU computing core for each offloading task, i.e., tasks offloaded on the same edge node have the same task-computing speed, and the number of offloading tasks that the edge node can host at the same time is related to the number of CPU cores. The energy consumption factor of the edge node is defined as $\varepsilon_o = 10^{-27}$ [34]. The delay ($T_{d_{ij,\text{comp}}}^{o}$) and energy consumption ($E_{t_{ij,\text{comp}}}^{o}$) of the edge node (e.j) in computing the offloading task ($t_{ij}^{o}$) are

$$T_{d_{ij,\text{comp}}}^{o} = \frac{D_{ij}^{o}}{f_{oj}}$$

$$E_{t_{ij,\text{comp}}}^{o} = \varepsilon_o D_{ij}^{o} (f_{oj})^{2}$$

In the block consensus verification scenario, the delay and energy consumption generated by block generation are not calculated in this paper because the overall overhead of block creation is small compared to that of block verification, where there is a large number of validation links, which has a low impact on the overall performance of the system. When the edge node performs block validation, assuming that the validation computation period of block $b$ is $D_{b}^{t}$, the validation delay ($T_{d_{ij,\text{comp}}}^{v}$) and energy consumption ($E_{t_{ij,\text{comp}}}^{v}$) of the blockchain validation node (e.v) are

$$T_{d_{ij,\text{comp}}}^{v} = \frac{D_{b}^{t}}{f_{ev}}$$

$$E_{t_{ij,\text{comp}}}^{v} = \varepsilon_o D_{b}^{t} (f_{ev})^{2}$$
3.3.3. Comprehensive Model

In this paper, we comprehensively calculate the delay and energy cost of the blockchain-based MEC task-offloading model by combining the designed communication and computation models.

(1) Latency Cost

The time delay in the quality of service model designed in this paper contains two links, namely task processing and block verification links. When calculating the time delay of the task processing link, it is assumed that all users start a local task and offload task processing from the same moment, i.e., local task computation and task offload transmission are carried out at the same time, so the actual time delay of task processing is the maximum value of the time delay of local computation and offload processing. The task-offloading delay consists of the communication delay ($T_{i,comm}^{d}$) of the user offloading the task to the edge node and the computation delay ($T_{i,comp}^{d}$) of the task on the edge node. If the user offloads the task to more than one edge node, the task-offloading delay is only computed for the longest processing link; then, the task offloading delay ($T_{i}^{d}$) of user $u_i$ is denoted as

$$T_{i}^{d} = \max\left(T_{i,comm}^{d_1}, \ldots, T_{i,comm}^{d_j}, \ldots, T_{i,comp}^{d_1}, \ldots, T_{i,comp}^{d_j}\right)$$

Furthermore, the task processing delay ($T_{i}^{d_i}$) of user $u_i$ is denoted as

$$T_{i}^{d_i} = \max(T_{i,comm}^{d_i}, T_{i,comp}^{d_i})$$

Similarly, when calculating the delay of the block verification link, since the packing node sends the block to each verification node for block verification at the same time, the block verification delay ($T_{i}^{d_v}$) is the maximum delay processed by each verification node and is denoted as

$$T_{i}^{d_v} = \max\left(T_{i,comm}^{d_1}, \ldots, T_{i,comm}^{d_j}, \ldots, T_{i,comp}^{d_1}, \ldots, T_{i,comp}^{d_j}\right)$$

In summary, the delay ($T_{i}^{d}$) of the quality of service model for user device $u_i$ in time slot $t$ is

$$T_{i}^{d} = T_{i}^{d_v} + T_{i}^{d_o}$$

(2) Energy Cost

In the energy consumption calculation process, the energy consumption of the communication model and the computation model are obtained by summing the processing energy consumption of each task.

Then, the communication and computation energy of user device $u_i$ in time slot $t$ are

$$E_{i,comm}^{l} = \sum_{e_j \in E} E_{i,j,comm}^{l} + \sum_{e_v \in EV} E_{i,v,comm}^{l}$$

$$E_{i,comp}^{l} = E_{i,comp}^{l} + \sum_{e_j \in E} E_{i,j,comp}^{l} + \sum_{ev \in EV} E_{i,v,comp}^{l}$$

In summary, the energy consumption ($E_{i}^{l}$) of the quality of service model for user device $u_i$ in time slot $t$ is

$$E_{i}^{l} = E_{i,comm}^{l} + E_{i,comp}^{l}$$

3.4. Incentive Reward Model

Previous research [22,43–45] has integrated the incentive mechanism of blockchain into the study of task pricing and resource allocation of MEC, balancing the allocation of edge service resources and value gains by considering game theory and auction theory. In this paper, the design of the incentive mechanism is simplified, and only the edge nodes participating in task-offloading computation and block verification are considered to be provided with incentive tokens in equal proportions according to energy consumption.
Hence, the incentive model favors edge nodes obtaining more incentive tokens to gain more benefits. In the incentive model, the blockchain uses the $\beta$ ratio of the unit of energy converted into obtainable tokens based on the energy consumption of the edge nodes; then, the tokens generated by agent $u_i$ are calculated as

$$ I^t_j = \left\{ \begin{array}{l} \beta \sum_{t \in E} \left( E_{ij,comp}^{t,o} + E_{ij,comp}^{t,d} \right), e_j \in V \\ \beta \sum_{e_j \in E} E_{ij,comp}^{t,o}, e_j \notin V \end{array} \right. $$

(23)

3.5. Privacy Model

In this section, we mainly consider that in the process of MEC task offloading, if we consider the energy consumption and delay factors of task communication and computation, user terminals often tend to offload a large number of tasks to edge nodes that are closer to them and have higher levels of resources. However, such a task-offloading method potentially risks data privacy leakage because MEC tasks usually contain sensitive private data such as the physical location of the device, identity characteristics, task data, etc. Suppose that many tasks containing private information are offloaded to an edge node. In that case, the edge node, out of its curiosity or due to being hijacked by an adversary, may collect and infer the user’s location and business characteristics based on the user’s offloading preferences. More seriously, the edge node may predict the user’s private information based on these data characteristics, resulting in user privacy leakage [46]. Therefore, it is necessary to design a privacy metric model to evaluate the degree of privacy leakage that may be caused by the user in the process of task offloading.

Information entropy is a concept that measures the uncertainty or amount of information. Privacy computing models can be utilized to assess and reduce privacy risks. The information entropy-based privacy measure is advantageous in the task of measuring the privacy leakage of user data and has been applied in research on MEC task offloading [41,47]. Therefore, this paper uses the privacy metric based on information entropy to measure the MEC task offloading privacy protection effect.

We define user $u_i$’s task-offloading preference ($P^t_i$) and measure the probability that user $u_i$’s data are exposed to edge nodes by calculating the ratio of user $u_i$’s offloaded task data volume to the total task data volume ($P^t_i$), which is calculated as follows:

$$ P^t_i = \frac{d^t_o}{d^t_i} = \frac{\sum_{t \in E} d^t_{ij,comp}}{d^t_i} $$

(24)

Based on the user’s task-offloading preference, the concept of privacy entropy is further adopted to describe the amount of privacy information carried by the offloading strategy of user $u_i$, $H^t_i$. When there is no task offloading on the user’s terminal, i.e., $P^t_i = 0$, the edge node cannot infer the user’s task information. The privacy entropy is at the maximum value ($H_{max}$), and in this paper, we set the value of maximum entropy to 10. The privacy entropy of user $u_i$ is calculated as

$$ H^t_i = \left\{ \begin{array}{l} -P^t_i \log_2 P^t_i, 0 < P^t_i < 1 \\ H_{max}, P^t_i = 0 \end{array} \right. $$

(25)

4. Problem Description

This paper’s optimization objectives for task offloading in mobile blockchain edge networks focus on privacy preservation, quality of service, and incentive reward. Privacy protection requires maximization of the privacy entropy of the privacy-preserving model to prevent users from offloading too much private data to the edge servers, leading to user privacy leakage. Quality user experience requires minimization of the latency and energy consumption of offloading user tasks. Incentive rewards require maximization of the workload of nodes in the blockchain edge network and improvement of the workload and efficiency of nodes. In this paper, by comprehensively considering offloading privacy,
Mathematics 2024, 12, 2264

quality of service, and incentive reward factors, the optimization problem can be formulated as the maximum value of the comprehensive optimization objective for user device $u_i$ and edge servers within time slot $t$ under the satisfaction of multiple constraints. The specific optimization objective function and constraints are expressed as follows:

$$P : \text{max } C_t^i = \omega_1 \ast H_t^i + \omega_2 \ast I_t^i - \omega_3 \ast Td_t^i - \omega_4 \ast En_t^i$$

(26)

Subject to:

$$s.t. Td_t^i \leq Td_{i,max}$$

(27)

$$0 \leq pw_{i,t} \leq pw_i$$

(28)

$$0 < P_t^i \leq 1$$

(29)

$$H_t^i \leq H_{max}$$

(30)

where $\omega_1, \omega_2, \omega_3, \text{ and } \omega_4$ are the weights of the indicators, which are used to specify the level of importance of different indicators. Equation (27) means the total task delay is constrained by the maximum tolerable delay of the task. Equation (28) means the device-to-node transmission power receives the constraint of the total transmission power. Equation (29) means the amount of offloaded task data of any user device does not exceed the constraint of the total task data. Equation (30) means the user’s privacy entropy is subject to the constraint of the maximum entropy value.

It is not difficult to find that the optimization problem presented in this paper is a mixed-integer linear programming problem, which are usually NP-hard and, therefore, difficult to solve with a globally optimal solution. The decision-making process for such problems occurs in a dynamic environment of long-term optimization, which makes it difficult for traditional convex optimization algorithms to adapt to unknown environments and perform adaptive optimization.

5. Algorithm

To address the environmental complexity and multi-objective competitiveness possessed by the above optimization problem description, this section first proposes an actor-critic deep reinforcement learning algorithm based on multiple agents sharing a global memory pool to improve the robustness and stability of performance. Secondly, the optimization problem is reformulated as a Markov process (MDP) by constructing each agent’s state space, action space, immediate rewards, and state transitions, and the algorithmic framework structure is described in detail.

5.1. Construction of the Markov Decision Process

In the blockchain mobile edge network task-offloading environment designed in this paper, each user device acts as a reinforcement learning agent, adopting a decentralized execution and centralized training model, which enables the agent to make independent decisions based on its observed and learned strategies. Multiple edge servers form a federated blockchain, sharing network parameters to jointly hold global information about the entire system. At the beginning of each time slot, user devices can initiate task processing requests, sending task and localization information to edge servers. After the edge server obtains the global network state information through blockchain sharing, it conducts centralized training. After training, each agent makes distributed local decisions based on its observations.

In order to solve the above optimization problem, it needs to be converted to the standard form of the Markov decision process (MDP) when using reinforcement learning algorithms. The key components of this transformation include defining the state space, action space, reward space, and state space transitions for each agent.

(1) State Space

The state space ($s_t^i$) of an agent ($i$) in time slot $t$ consists of the localization ($loc_t^i = (x_t^i, y_t^i)$) of its corresponding user device ($u_i$) and the amount of requested task...
data \((d_i^t)\), i.e., \(s_i^t = (l_i^t, d_i^t)\). Therefore, the state space \((s^t)\) of the reinforcement learning algorithm as a whole is denoted as \(s^t = (s_1^t, \ldots, s_n^t)\).

(2) **Action space**

The action space \((a_i^t)\) of agent \(i\) in time slot \(t\) represents the distribution of request data processing and channel power allocation of user device \(u_i\) in the current network state, i.e., \(a_i^t = (d_i^{t^d}, d_i^{t^a}, \ldots, d_i^{t^m}, p_i^{t}, \ldots, p_i^{t_n})\).

(3) **Reward function**

The reward function of the blockchain mobile edge network task-offloading model aims to maximize the optimization objective function \((C_i^t)\) of each agent, i.e., maximize the privacy entropy of the user device to safeguard the privacy of user data, as well as the blockchain rewards computed by completing the offloaded tasks, and, at the same time, minimize the task processing latency and energy consumption of the user device in order to provide the user with a higher quality of service. The reward function at time slot \(t\) is expressed as follows:

\[
r_i^t = \begin{cases} \omega_1 * H_i^t + \omega_2 * I_i^t - \omega_3 * Td_i^t - \omega_4 * En_i^t, & \text{Equations (27)-(30)} \\ r_0, & \text{other} \end{cases}
\]

where \(r_0\) is a constant much smaller than 0 that represents the value of the algorithmic base reward given by the environment if the current policy does not satisfy the constraints of Equations (27)-(30).

5.2. **Algorithmic Framework**

The framework of the algorithm proposed in this paper is shown in Figure 3. The algorithm sets a corresponding agent for each user device, including an actor network, a critic network, and a random sampler. The actor network and critic network adopt a dual neural network structure. The current network is responsible for constructing the actor’s policy network \((\pi_i)\) and the critic’s value network \((Q_i)\). The \(Q\) value of the critic network represents the expected reward for taking a particular action in a given state. The target network is softly updated using the current network parameters \((\theta_{\pi_i}^t\) and \(\theta_{Q_i}^t)\), thus guaranteeing the stability of network learning.

We assume that the sample value function for the critic target network to compute time slot \(t\) is \(Q_i(s_i^t, a_i^t|\theta_{Q_i}^t)\); then, the target \(Q\) value can be calculated as

\[
q_i = r_i^t + \gamma Q_i(s_i^{t+1}, a_i^{t+1}|\theta_{Q_i}^t),
\]

where \(\gamma\) denotes the discount factor.

To update the critic’s current network parameter \((\theta_{Q_i}^t)\), the loss values of the parameters are computed using a mean-square error function. The mean-square error function can help the critic network accurately predict the value of a state or state–action pair.

\[
\text{Loss}(Q_i) = E[(Q_i(s_i^t, a_i^t|\theta_{Q_i}^t) - q_i)^2] = \frac{1}{n} \sum_{i=1}^{n} (Q_i(s_i^t, a_i^t|\theta_{Q_i}^t) - q_i)^2
\]

We minimize \(\text{Loss}(\theta_{Q_i}^t)\) by gradient descent, and the update method for the \(\theta_{Q_i}^t\) parameter is denoted by

\[
\theta_{Q_i}^t \leftarrow \theta_{Q_i}^t + \alpha \nabla_{\theta_{Q_i}} \text{Loss}(Q_i),
\]

where \(\alpha\) is the learning rate of the critic’s current network parameter \((\theta_{Q_i}^t)\).
Figure 3. Algorithm structure.

The actor network constructs the action policy ($\pi_i$) based on the state space ($s^t_i$) of the reinforcement learning agent in time slot $t$ and the reward function ($r^t_i$) and generates the action ($a^t_i$) in the time slot, which can be represented as

$$a^t_i = \pi_i(s^t_i | \theta^\pi_i)$$

(35)

However, using the output of the strategy network directly does not allow the agent to discover more strategies, so an exploration strategy needs to be constructed by adding noise.

$$a^t_i = \pi_i(s^t_i | \theta^\pi_i) + \tau N_t$$

(36)

where $\tau$ denotes the attenuation factor of the noise, which gradually decreases with the number of iterations of the algorithm to guarantee the stability of network training and $N_t$ is Gaussian noise obeying a normal random distribution.

The policy objective function of the actor network is

$$J(\pi_i) = E[Q_i(s^t_i, a^t_i | \theta^Q_i)]$$

(37)

Then, the gradient of the objective function of the strategy is expressed as

$$\nabla_{\theta^\pi_i} J(\pi_i) = E[\nabla_{a^t_i} Q_i(s^t_i, a^t_i | \theta^Q_i) \nabla_{\theta^\pi_i} \pi(s^t_i | \theta^\pi_i)]$$

(38)

Then, the update method for the $\theta^\pi_i$ parameter is expressed as

$$\theta^\pi_i \leftarrow \theta^\pi_i + \beta \nabla_{\theta^\pi_i} J(\pi_i)$$

(39)

where $\beta$ is the learning rate of the actor network’s $\theta^\pi_i$ parameter.

In addition, the soft update method for the actor and critic target network parameters ($\theta^\pi_i'$ and $\theta^Q_i'$) can be represented as

$$\theta^\pi_i' \leftarrow \sigma \theta^\pi_i + (1 - \sigma) \theta^\pi_i$$

(40)

$$\theta^Q_i' \leftarrow \sigma \theta^Q_i + (1 - \sigma) \theta^Q_i$$

(41)

where $\sigma \in (0, 1)$ is the soft update weight.
In order to reduce environmental changes due to policy learning by other agents, this paper adopts a global memory pool to store the experience samples \((s_i^t, s_i^{t+1}, a_i^t, r_i^t)\) of each agent and uses it to train the neural network of the agents. The global memory pool can be constructed by using the blockchain to realize the sharing of information among agents in the actual application process.

In order to better understand the idea and process of this paper, the pseudo-code of the algorithm is shown in Algorithm 1.

**Algorithm 1: Actor–Critic Algorithm for Blockchain–MEC Task Offloading**

**Data:** Blockchain-MEC environment parameters, user mobile device states

**Result:** Task offloading strategies for each user mobile device

1. for agent \(i \in [1, N]\) do
   2.   // initialize algorithm parameters
   3.   initialize actor current network \(\pi_i^t\) with \(\theta_{\pi i}^t\)
   4.   initialize critic current network \(Q_i^t\) with \(\theta_{Q i}^t\)
   5.   initialize actor and critic target parameter \(\theta_{\pi i}^{t\tau} \leftarrow \theta_{\pi i}^t\) and \(\theta_{Q i}^{t\tau} \leftarrow \theta_{Q i}^t\)
   6.   clear global memory pool
   7. end

8. for iteration \(\in [1, \text{max\_iter}]\) do
   9.   each agent initializes initial state space \(s_i^0\)
10. for \(t \in [1, T]\) do
   11.     for agent \(i \in [1, N]\) do
   12.       // select action space
   13.       get \(a_i^t \leftarrow \pi_i(s_i^t|\theta_{\pi i}^t)\) by action current network
   14.       // get state space
   15.       get \((s_i^{t+1}, r_i^t) \leftarrow \text{env}(a_i^t)\) by environment
   16.       // update global memory pool
   17.       push \((s_i^t, s_i^{t+1}, a_i^t, r_i^t)\) in the memory pool
   18.       // update state space
   19.       \(s_i^t \leftarrow s_i^{t+1}\)
   20.       get \(M\) samples from memory pool
   21.       compute \(y_i = r_i^t + \gamma Q_i(s_i^{t+1}, a_i^{t+1}|\theta_{Q i}^{t\tau})\)
   22.       // calculate parameter gradient and update parameters
   23.       \(\theta_{Q i}^t \leftarrow \theta_{Q i}^t + \alpha \nabla_{\theta_{Q i}^t} \text{Loss}(Q_i)\)
   24.       \(\theta_{\pi i}^t \leftarrow \theta_{\pi i}^t + \beta \nabla_{\theta_{\pi i}^t} J(\pi_i)\)
   25.       // soft update Actor and Critic network parameters
   26.       \(\theta_{Q i}^{t\tau} \leftarrow \delta \theta_{Q i}^t + (1 - \delta)\theta_{Q i}^{t\tau}\)
   27.       \(\theta_{\pi i}^{t\tau} \leftarrow \delta \theta_{\pi i}^t + (1 - \delta)\theta_{\pi i}^{t\tau}\)
   28.     end
   29. end
   30. return each device’s optimal migration strategy \(a_i^*\) and the minimum total target cost \(C^*\)

5.3. Complexity Analysis

In this paper, the computational complexity of the proposed algorithm is mainly considered to be the sum of the training time overhead of all the agents. We assume that \(n\) is the number of agents, \(L_a\) is the number of neural network layers of the actor network, \(L_c\) is the number of neural network layers of the critic network, \(S\) is the number of samples of each agent from the global memory pool, \(I\) is the number of algorithmic iterations, \(d_s\) is
the state-space dimension, and $d_a$ is the action-space dimension. Then, the computational complexity of the algorithm can be calculated as $O(nSI(L_a + L_c)(d_s + d_a)^2)$.

6. Experiment and Discussion

In this section, our proposed algorithm is evaluated and analyzed through simulation experiments.

6.1. Experimental Environment

The hardware and software specifications of the experimental environment described in this paper are shown in Table 2.

**Table 2. Hardware and software specifications.**

<table>
<thead>
<tr>
<th>Designation</th>
<th>Specification Version</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>AMD Ryzen 7-5800</td>
</tr>
<tr>
<td>GPU</td>
<td>Nvidia RTX3060</td>
</tr>
<tr>
<td>Memory</td>
<td>80 GB RAM</td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td></td>
</tr>
<tr>
<td>Operation system</td>
<td>Windows11</td>
</tr>
<tr>
<td>Language</td>
<td>python3.7.16</td>
</tr>
<tr>
<td>Deep learning framework</td>
<td>torch1.10.0</td>
</tr>
<tr>
<td>Library function</td>
<td>numpy1.21.6</td>
</tr>
</tbody>
</table>

6.2. Parameter Design

In order to realize the simulation of the network model, this paper simulates the mobile user task-offloading environment in real scenarios in a $1000 \times 1000$ area (Figure 4) that contains four blockchain–MEC servers at fixed locations and user mobile devices moving along the path of black arrows. The servers receive task offload requests from user mobile devices and specify the user offload policy for the devices through collaborative planning using multiple servers. The user’s mobile device moves along the non-random irregular black arrow path with a fixed step size in each time slot. It generates a random amount of task data, which are offloaded to one or more servers for processing according to the task-offloading policy.

![Figure 4. Network environment simulation.](image-url)
The parameters of the reinforcement learning algorithm and blockchain edge network environment are shown in the following Table 3.

<table>
<thead>
<tr>
<th>Parameter Kind</th>
<th>Parameter Symbol</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model parameters</td>
<td>$d_s$</td>
<td>State-space dimension</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$d_a$</td>
<td>Action-space dimension</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>$\delta$</td>
<td>Soft update weights</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>Critic update parameters</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>$\beta$</td>
<td>Actor update parameters</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>$[\omega_1, \omega_2, \omega_3, \omega_4]$</td>
<td>Reward function weights</td>
<td>$[0.8, 0.09, 0.09, 0.02]$</td>
</tr>
<tr>
<td>Environmental parameters</td>
<td>$[\text{loc}_1, \text{loc}_2, \text{loc}_3, \text{loc}_4]$</td>
<td>Edge node location</td>
<td>(333, 333), (333, 666), (666, 333), (666, 666)</td>
</tr>
<tr>
<td></td>
<td>$p_{\text{uw}_i}$</td>
<td>User terminal transmission power</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>$p_{\text{nw}_e}$</td>
<td>Edge node transmission power</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$f_{\text{ul}}$</td>
<td>User terminal processing frequency</td>
<td>$10 \times 10^8$</td>
</tr>
<tr>
<td></td>
<td>$f_{\text{ue}}$</td>
<td>Edge node processing frequency</td>
<td>$4 \times 10 \times 10^8$</td>
</tr>
<tr>
<td></td>
<td>$d_0$</td>
<td>Block header size</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$N_0$</td>
<td>Wireless channel noise</td>
<td>$10 \times 10^{-7}$</td>
</tr>
<tr>
<td></td>
<td>$B$</td>
<td>Wireless communication bandwidth</td>
<td>$3 \times 10 \times 10^8$</td>
</tr>
<tr>
<td></td>
<td>$R$</td>
<td>Server wired communication rate</td>
<td>10</td>
</tr>
</tbody>
</table>

### 6.3. Experimental Analysis

#### 6.3.1. Contrasted Algorithms

- **JODRL-PP [33]**: The JODRL-PP (Joint Optimal Deep Reinforcement Learning with Privacy Preservation) algorithm is a stochastic game-theoretically based task-offloading problem for multi-access point environments proposed for multi-agent deep reinforcement learning algorithms. The algorithm uses a trusted third party for centralized training. It achieves distributed execution to improve the quality of the results while considering the dynamic changes in a multi-user environment and dealing with the complexity of multiple users and access points through stochastic game theory.

- **IQL [48]**: IQL (Independent Q-Learning) is a reinforcement learning algorithm applied in multi-agent systems. In a multi-agent system, each agent learns its own Q-value function independently without considering the actions and strategies of other agents and uses only its own state and action information in the learning process. In the IQL-based task-offloading algorithm, if an agent does not cache the corresponding requested service, the agent migrates the task to be executed to another agent that has cached the service based on the service cache information shared among the agents at the beginning of each time slot.

- **Qmix [49]**: QMIX (Q-value Mixing Network) is a value-based multi-agent reinforcement learning algorithm that can be used to train decentralized policies in a centralized end-to-end manner. In addition, QMIX’s network estimates joint action values as complex nonlinear combinations of per-agent values conditional only on local observations. It requires that the joint action values for each agent be monotonic. This maximizes the joint action values that can be handled in non-strategy learning and ensures consistency between centralized and decentralized strategies.

- **VDN [50]**: The VDN (Value-Decomposition Network) is a value decomposition method for multi-agent systems that decomposes the global value function into local...
value functions. Each agent learns only the local value function associated with it. This network architecture learns to decompose the team value function into the value functions of agents. It solves the problem of collaborative reinforcement learning of multiple agents with a single joint reward signal. The VDN algorithm does not consider the spatial relationship of the type of service request and the state of the wireless network among agents, and it directly decomposes the joint action value function into the sum of the local action value functions of all agents.

6.3.2. Results

(1) Experiment 1: Performance Comparison

We set up ten random mobile users in the experimental simulation environment by recording the reward function during 1000 iterations of the reinforcement learning algorithm, the result of which is shown in Figure 5. From the figure, we can find that compared with other schemes, the proposed algorithm’s curve of the final stabilization reward function value is significantly higher than that of other algorithms, and the fluctuation amplitude after stabilization is smaller.

![Figure 5. Reward function value iteration.](image)

In order to minimize the impact of single-experiment error on the results, we conducted five repetitive experiments. We recorded the average reward function values for different algorithm configurations for all training cycles, and the comparison results are shown in Figure 6. From the figure, we can find that the proposed algorithm improves by more than 40% in performance compared to QMIX, IQL, and VDN and outperforms JODRL-PP, indicating that the proposed algorithm can obtain a better solution to the problem set in this paper.

In our experiments, we also recorded the average costs of task processing energy consumption, task processing latency, user privacy metrics, and blockchain incentive rewards in the reward function, and the comparison graphs are shown in Figure 7. Through the comparison, we can find that the proposed algorithm significantly outperforms QMIX, IQL, and VDN in all costs except blockchain incentive rewards, except that the proposed algorithm reduces the energy cost by 44.38% and improves the blockchain incentive rewards...
by 13.27% compared to the JODRL-PP algorithm. However, the proposed algorithm is inferior in terms of task processing latency and user privacy metrics.

Figure 6. Average reward function value comparison.

Figure 7. (a) Average processing delay comparison; (b) average energy consumption comparison; (c) average incentive reward comparison; (d) average privacy metric comparison.
(2) Experiment 2: Performance Comparison under Different User Scales

In order to test the changes of the algorithms in the optimization problem proposed in this paper under different user sizes, we set the user sizes to 10, 15, 20, 25, and 30 and recorded the average reward function values of the algorithms under different user sizes in five groups of repeated experiments. The results are shown in Figure 8. From the figure, we can find that with increasing user size, the average reward function value of all models decreases; this is because with the increase in users, the corresponding amount of user tasks is also raised. The delay and energy consumption required to process the task increase due to the existence of an upper limit of the user’s privacy metric, and the blockchain network incentive rewards are subject to the limitation of the amount of nodes to receive the task. Hence, a decrease in the value of the reward function is a normal phenomenon. The proposed algorithm still has an optimal average reward function value based on different agent scales.

![Figure 8. Reward function value iteration for different user scales.](image)

The experimental comparison graphs of the average cost of task processing energy consumption, task processing delay, user privacy metrics, and blockchain incentive rewards are shown in Figure 9. The proposed algorithm has advantages in some single cost metrics in growing user size, and the experimental results are similar to those of Experiment 1.

(3) Experiment 3: Ablation Experiment

In this paper, we design ablation experiments to investigate the effects of Gaussian noise-based action-space search in the proposed algorithm and the global memory pool of agents on the performance of the algorithm. As in Experiment 1, we set up 10 random mobile users in the experimental simulation environment by recording the reward function value during 1000 iterations of the reinforcement learning algorithm, and the result is shown in Figure 10. From the figure, we can find that the curve of the proposed algorithm reaches a stabilization level faster than that of the other two configurations. It exhibits less fluctuation of the state after stabilization. In addition, the algorithm’s final stabilization reward function value is significantly higher than that of the other two configurations, which indicates that the algorithm’s overall performance has been improved.
Figure 9. (a) Average processing delay comparison; (b) average energy consumption comparison; (c) average incentive reward comparison; (d) average privacy metric comparison for different agent scales.

Figure 10. Reward function value iteration.

In addition, we compared the average rewards of different algorithm configurations through five repetitions of the experiment, as shown in Figure 11. From the figure, it can be seen that the average reward function value of the proposed algorithm possesses a significant advantage throughout the training cycle. The performance is improved
by 38.36% and 43.59% compared to the schemes lacking Gaussian process action-space selection noise and global memory pool, respectively.

Figure 11. Average reward function value comparison.

According to the results of the ablation experiments, the introduction of Gaussian noise-based action-space search and global shared memory pool significantly improved the algorithm’s performance. These two improvements enhance the algorithm’s ability to explore and utilize historical information, thus improving the learning efficiency and quality of the policy in the long run. This enhancement is significant in complex and dynamic environments, requiring the algorithm to adapt and discover new and better strategies quickly.

In summary, the proposed algorithm was analyzed and validated through many comparative experiments, and we demonstrated the advantages of the proposed algorithm over comparative algorithms in terms of global optimization objectives. Through ablation experiments, we analyzed the important role of Gaussian noise-based action-space search and global shared memory pooling. However, the proposed algorithm still has a disadvantage in calculating task processing delay cost.

7. Conclusions and Future Works

In this paper, we propose a blockchain-based MEC task offloading strategy based on multi-agent reinforcement learning that utilizes a global memory pool to enable each agent to acquire the experience of other agents during the training process in order to enhance the collaborative ability among agents and the overall performance of the system. Moreover, the algorithm introduces a search strategy based on decodable Gaussian random noise action space, improving the agents’ search state space to avoid falling into the local optimum. In terms of the optimization objective function, this paper comprehensively considers cost factors such as task execution energy consumption, processing delay, user privacy metrics, and blockchain incentive rewards and innovatively proposes a blockchain-based MEC task-offloading model. The experimental results show that compared with other algorithms, the proposed algorithm improves the performance of the global optimization objective by more than 10% and has obvious advantages in energy consumption and blockchain incentive rewards. In addition, the ablation experiments show that the Gaussian process
action-space selection noise and the global memory pool improve the performance by 38.36% and 43.59%, respectively.

However, this paper is subject to limitation in terms of problem modeling and algorithm design. Firstly, we only used the existing consensus mechanism and simplified incentive mechanism to simulate the execution process of blockchain on MEC, which still has a large deviation from the actual scenario. Secondly, we must consider more security elements of MEC task offloading in the model design. Thirdly, we still need to improve the algorithm’s execution efficiency. Therefore, further research and optimization of problem modeling and algorithm design for blockchain-based MEC task offloading are important research directions for us in the future.

Author Contributions: C.L. developed the idea, performed research and analyses, and wrote the manuscript. Z.S. verified and revised the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China (No. 62272239), the Postgraduate Research & Innovation Plan of Jiangsu Province (No. KYCX20_0761), and the Jiangsu Agriculture Science and Technology Innovation Fund (No. CX(22)1007).

Data Availability Statement: Data are contained within the article.

Acknowledgments: We wish to thank all code providers. We also wish to thank all colleagues, reviewers, and editors who provided valuable suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

References


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.