sensors

MDPI

*Article*

# Joint Optimization for Mobile Edge Computing-Enabled Blockchain Systems: A Deep Reinforcement Learning Approach

**Zhuoer Hu [1], Hui Gao [1,*], Taotao Wang [2], Daoqi Han [1] and Yueming Lu [1]**

[1] Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China; zhuoer_hu@bupt.edu.cn (Z.H.); handq@bupt.edu.cn (D.H.); ymlu@bupt.edu.cn (Y.L.)
[2] College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China; ttwang@szu.edu.cn
* Correspondence: huigao@bupt.edu.cn

**Abstract:** A mobile edge computing (MEC)-enabled blockchain system is proposed in this study for secure data storage and sharing in internet of things (IoT) networks, with the MEC acting as an overlay system to provide dynamic computation offloading services. Considering latency-critical, resource-limited, and dynamic IoT scenarios, an adaptive system resource allocation and computation offloading scheme is designed to optimize the scalability performance for MEC-enabled blockchain systems, wherein the scalability is quantified as MEC computational efficiency and blockchain system throughput. Specifically, we jointly optimize the computation offloading policy and block generation strategy to maximize the scalability of MEC-enabled blockchain systems and meanwhile guarantee data security and system efficiency. In contrast to existing works that ignore frequent user movement and dynamic task requirements in IoT networks, the joint performance optimization scheme is formulated as a Markov decision process (MDP). Furthermore, we design a deep deterministic policy gradient (DDPG)-based algorithm to solve the MDP problem and define the multiple and variable number of consecutive time slots as a decision epoch to conduct model training. Specifically, DDPG can solve an MDP problem with a continuous action space and it only requires a straightforward actor–critic architecture, making it suitable for tackling the dynamics and complexity of the MEC-enabled blockchain system. As demonstrated by simulations, the proposed scheme can achieve performance improvements over the deep Q network (DQN)-based scheme and some other greedy schemes in terms of long-term transactional throughput.

**Keywords:** blockchain; mobile edge computing; computation offloading; deep deterministic policy gradient (DDPG)



**Citation:** Hu, Z.; Gao, H.; Wang, T.; Han, D.; Lu, Y. Joint Optimization for Mobile Edge Computing-Enabled Blockchain Systems: A Deep Reinforcement Learning Approach. *Sensors* **2022**, *22*, 3217. https://doi.org/10.3390/s22093217

Academic Editors: Hao Wang, Małgorzata Kujawska, Vijayakumar Varadarajan, Han-Chieh Chao, Lidia Dobrescu, Sheng-Lyang Jang, Yi Wu, Wen-Cheng Lai, Adam W. Skorek and Rashmi Bhardwaj

Received: 9 February 2022
Accepted: 19 April 2022
Published: 22 April 2022



## 1. Introduction

With the wide utilization of intelligent mobile devices in the fifth-generation (5G) era and the advances in wireless communication techniques in the forthcoming sixth-generation (6G) communication systems [1], the internet of things (IoT) is increasingly attracting attention from both academia and industry as a versatile technology [2,3]. Nowadays, more and more intelligent devices get access to IoT networks, where each object with identifying, sensing, networking, and processing capabilities can communicate with other nodes. Such ubiquitous interconnections generate massive data to be stored, processed, and analyzed [4,5].

Recent advances in information and communication technologies are accelerating the IoT's transition to the 6G era. As a result, new IoT infrastructures and data processing architectures are under construction. Currently, a majority of IoT applications depend on centralized cloud servers to store data and process tasks [6], which necessitates that the third parties owning the cloud servers are quite trustworthy, otherwise the user data may be exposed to security concerns [7]. Furthermore, centralized cloud-based applications

introduce delay and privacy issues [8,9]. Therefore, demands for data security, processing efficiency, and low operational cost are overgrowing in IoT scenarios [10]. Blockchain [11], as a decentralized data storage technology, can guarantee that each transaction record is immutable through an encryption algorithm and distributed structure. As a result, blockchain has been presented as a promising technique for enhancing the security and efficiency of data storing/fetching in IoT networks, which can realize tamper resistance and data availability for IoT networks in a decentralized manner [12–14].

In the current IoT architecture, blockchain technology has been extensively investigated [15]. Kang et al. have proposed a blockchain-enabled internet of vehicles framework based on an upgraded delegated proof-of-stake (DPoS) consensus mechanism rather than the native proof-of-work (PoW) or proof-of-stake (PoS) mechanism to improve the security of vehicle data sharing [16]. Fan et al. presented a blockchain-based strategy for resolving the security issue associated with time synchronization in IoT networks [17]. W. Li et al. proposed a data security strategy based on blockchain technology for intelligent applications in 6G systems [18]. However, these schemes focus on the security of block verification and block generation while lacking the consideration of emerging applications and services in IoT networks that require high computational workloads and low latency. Most existing blockchain-enabled frameworks cannot meet the demand for high transactional throughput in IoT scenarios with compute-intensive and real-time applications and services. Therefore, service-oriented blockchain-enabled frameworks are called for to meet the transactional throughput demands of current and next-generation IoT networks.

New IoT services and applications, such as mobile multimedia, visual sensors, smart grids, and intelligent vehicles, are computationally intensive and sensitive to latency, challenging the blockchain-enabled IoT framework design. Conventional cloud computing systems are confronted with the problems of long latency and overload problems, hindering the blockchain deployment in IoT networks. To address the above challenges, the mobile edge computing (MEC) [19] technique is considered a potential option. Because an MEC system is deployed at the edge of IoT networks and near the access network, it is capable of bridging the divide between the limited resources in the proximity of users and the ever-increasing computational demand of IoT applications [20]. Therefore, MEC technology can facilitate the development of low-latency, scalable, and blockchain-enabled IoT networks. Recently, several studies [21,22] have been proposed to enhance data security and transactional throughput for IoT by integrating blockchain technology and MEC technology. However, most existing schemes focus only on the computational offloading strategy of MEC or the working mechanism of blockchain and lack a comprehensive and specific analysis of the MEC-enabled blockchain system. Therefore, they fail to achieve a joint performance optimization, which leaves room to improve.

The efficient deployment and optimization of MEC-enabled blockchain systems in IoT networks are challenging from several perspectives: trade-off between latency and security, joint optimization, and dynamic continuous domain-based optimization. (1) *Trade-off between latency and security*: the blockchain-enabled IoT network is confronted with latency-sensitive challenges and thus requires an efficient blockchain mechanism without compromising security. Additionally, due to system resource limitations [23] in IoT networks, the collaborative design of an efficient resource allocation policy and a lightweight blockchain consensus mechanism at the edge of wireless networks is challenging. (2) *Joint optimization*: the optimization problem formulation in most existing studies [21,22] considers the blockchain system and MEC system separately, ignoring the coupling relationship between the blockchain transactional throughput and the MEC computation rate. Due to the lack of joint optimization consideration, the performance in existing studies can be improved. (3) *Dynamic continuous domain-based optimization*: the IoT services arrive in complicated stochastic patterns, and most computation tasks of IoT services have sensitive latency requirements, which also pose significant challenges to the joint optimization of MEC-enabled blockchain systems. Additionally, the parameters of MEC computation offloading policy or blockchain optimization strategy are continuous domain variables,

while existing works [21,22] simplify them to discrete variables. Therefore, we resort to continuous domain-based DRL to realize dynamic and continuous joint control of computation resource allocation and block generation.

To address the aforementioned issues, we present a DRL-based joint performance optimization framework for MEC-enabled blockchain systems in IoT networks, which aims to improve the scalability/throughput while guaranteeing data security and transaction processing efficiency. In particular, each IoT node can offload a portion of computation tasks to MEC servers for efficient data processing, wherein the computation tasks include the IoT application tasks and the tasks for block generation and reaching consensus. Meanwhile, blockchain technology is adopted for secure data storage and sharing inside this framework, with a consensus mechanism based on practical Byzantine fault tolerance (PBFT) and DPoS [24] being adopted. Furthermore, the performance optimization of MEC and blockchain system is jointly formulated as a Markov decision process (MDP) problem, where state transitions mainly depend on changes in time-varying factors such as the impact of user movement on wireless transmissions, node workload, etc, which are unknown a priori. Moreover, the action is selected based on continuous space. As a result, conventional math models are ineffective in solving the MDP problem. To address this MDP problem, a novel DRL-based algorithm with continuous action space is developed. It shows superiority in tackling dynamic and complicated joint optimization problems. The primary contributions of this paper are listed as follows:

(1) A novel MEC-enabled blockchain framework in IoT networks is developed, considering the latency and scalability issues that arose from the throughput requirements of future wireless networks and blockchain systems. We analyze MEC computation efficiency and critical performance indicators of blockchain, i.e., decentralization, latency, throughput, and adversarial fraction, which can guide the joint optimization of the framework.

(2) A novel MEC and blockchain joint optimization algorithm is developed for maximizing the computational efficiency of MEC and the transaction throughput of blockchain systems, which is formulated as an MDP problem. In contrast to most existing research [15–17] in which the modeling and optimization of MEC and blockchain systems are carried out independently, the block interval, block size, data transaction throughput, power allocation for local execution and task offloading, latency, and security constraints are jointly considered in the proposed algorithm. Therefore, we propose a more comprehensive scheme and address the blockchain deployment challenges in IoT scenarios.

(3) The MDP problem is solved using a deep deterministic policy gradient (DDPG)-based learning algorithm to tackle the dynamic and large-dimensional properties of IoT networks that are intractable using classic learning approaches such as $\mathcal{Q}$-learning [25]. In particular, the DDPG-based algorithm enables the joint resource allocation for MEC and blockchain systems in a continuous domain so as to solve the MDP problem with better convergence.

(4) Extensive simulation findings demonstrate that the presented performance optimization framework has the capacity to enhance the transaction processing efficiency of MEC-enabled blockchain networks significantly. The superiority of the DDPG-based algorithm over the deep Q network (DQN)-based algorithm [26] and other conventional schemes is verified.

The remainder of this paper is structured as follows. Section 2 introduces the related works. Section 3 provides the preliminaries of blockchain technology and a basic introduction of the consensus protocol based on DPoS and PBFT. Section 4 describes the system model. In Section 5, the DDPG-based joint performance optimization framework is proposed, wherein the joint optimization problem is formulated and solved using a DDPG-based approach. Section 6 evaluates the proposed algorithm in detail and discusses the simulation results. At last, Section 7 summarizes this paper and looks forward to future work.

## 2. Related Works

In this section, we first introduce the existing works on blockchain-enabled IoT networks. Then, existing works which utilize the MEC technology to improve the transaction throughput of blockchain-enabled IoT networks are introduced, and the main distinguishments between our proposed scheme and closely related works are discussed.

### 2.1. Blockchain-Enabled IoT Networks

Due to the properties of blockchain technology such as stability, privacy protection, and security, integrating blockchain technology into IoT systems for supporting IoT future development has received considerable attention recently [27]. Yang et al. present a blockchain-based system for IoT devices and a tailored smart contract to enable the holistic transactive energy management [28]. Nguyen et al. design a blockchain-based model for IoT data trading which ensures security and privacy [29]. Although blockchain provides security and privacy benefits for IoT networks, there remains a barrier to its adoption in IoT networks due to the resource constraints of IoT devices.

### 2.2. MEC-Enabled Blockchain for IoT Networks

MEC technology has been widely utilized to perform massive, parallel, and complex computations [30]. The MEC system enables resource-constrained IoT devices to offload computing tasks to edge MEC servers, therefore resolving the resource-intensive issues that blockchain-enabled IoT networks confront. There are some works that use the MEC technology to improve the transaction throughput of blockchain-enabled IoT networks, which are closely related to our study. Zhao et al. proposes a computation resource allocation strategy of a public blockchain network in MEC systems [21], which is designed for a one-time slot and would involve a huge computation cost for long-term performance optimization that edge devices cannot afford. Nguyen et al. presents a secure deep reinforcement learning (DRL)-based computation offloading approach and a trustworthy blockchain access control mechanism for the mobile blockchain-based IoT networks [22]. Qiu et al. develops an adaptive genetic algorithm (AGA)-based computation offloading scheme for MEC-enabled blockchain systems [31]. Guo et al. proposes a resource allocation and block generation scheme based on a double-dueling DQN algorithm [32]. However, both [22,31] perform the optimization of blockchain and MEC systems independently, resulting in inferior performance. Additionally, both [22,32] are discrete action space-based and incapable of handling continuous action cases. Therefore, we develop a DDPG-based joint performance optimization algorithm for blockchain and MEC systems while considering the decentralization, security, latency, and power consumption constraints.

In summary, Table 1 shows distinguishments between our proposed scheme and some existing research. Specifically, for the 'Blockchain Agent' column, we can find that the deployment of the blockchain agent in the proposed scheme differs from [31,32], which results in distinct system model designs. For the 'State Space Design' column, *A* considers system offloading cost, system computation resources, and system bandwidth resources; *B* considers task queue length, computation resources of users, network identification, and available server resources; *C* considers channel condition between users and BSs, channel condition between different BSs, computation resources of BSs, primary node; *D* considers channel condition between users and BSs, computation resources of users, task buffer of users, and stake distribution. All in all, the blockchain agent deployment and state space design in this paper differ from existing ML-based schemes and therefore leads to different system model and problem formulation.

**Table 1.** A comparison with related works.

| Work | Long-Term Reward | Joint Optimization | Blockchain Agent | State Space Design | ML Approach |
|---|---|---|---|---|---|
| [21] | × | × | × | / | × |
| [22] | ✓ | × | × | *A* | DQN |
| [31] | ✓ | × | IoT Community | *B* | AGA |
| [32] | ✓ | ✓ | Base Station (BS) | *C* | Double-Dueling DQN |
| Our proposed scheme | ✓ | ✓ | IoT Device | *D* | DDPG |

## 3. Preliminaries on Blockchain

This section will begin with an overview of blockchain technology, followed by an introduction to the fundamentals of the consensus protocol based on DPoS and PBFT.

### *3.1. Overview of Blockchain*

A blockchain is an encrypted and distributed database that maintains an ordered collection of transaction records. Blockchain has many security performance advantages over traditional centralized databases [33]. Firstly, the primary objective of blockchain technology is to prevent the database's data from being maliciously tampered with or stolen. Furthermore, the decentralized nature of the blockchain system distributes and stores blocks that contain transaction records across a large number of network nodes during the block generation process. Due to the fact that the block generation process involves a large number of network nodes, data transactions are only recorded on a block after it has been confirmed by other nodes. Therefore, data storage is not dependent on a single centralized database. As a result, blockchain technology outperforms traditional centralized databases in terms of reliability and resistance to the single point of failure attacks.

The general architecture of blockchain technology consists of six layers [34] as shown in Figure 1. A blockchain-enabled system is comprised of the following components: an encryption method, peer-to-peer (P2P) transmission, a consensus protocol, a distributed ledger, a smart contract, and the application scenario [35]. In particular, the consensus protocol is a critical component of the blockchain. A blockchain system based on the consensus protocol can validate data transaction records without the involvement of a trusted third party. In the following, the consensus protocol adopted in this study will be introduced.

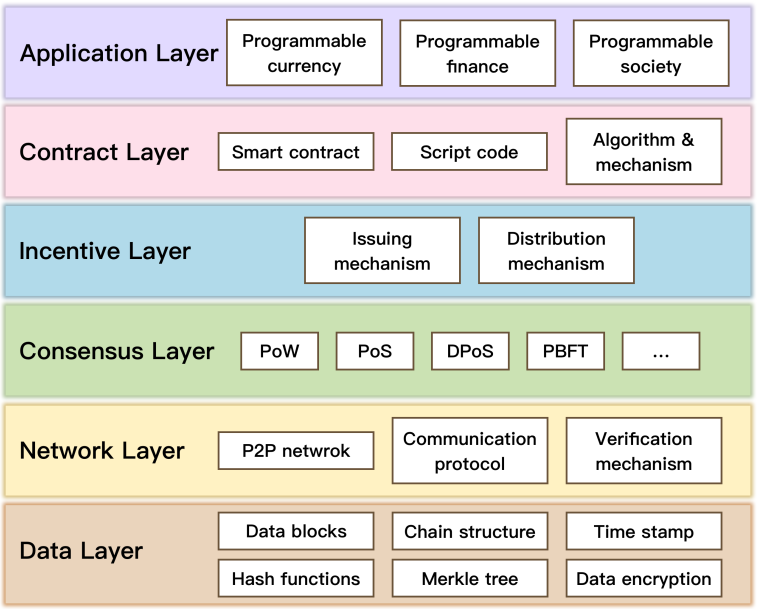


**Figure 1.** General architecture of blockchain.

### 3.2. Consensus Protocol Based on DPoS and PBFT

This subsection focuses on the consensus protocol, which is an important component of the blockchain system. Different blockchain systems handle the consensus problem by adopting different protocols that require validators to demonstrate their neutrality [36]. The most widely used consensus protocols include PoW, PoS, DPoS, and PBFT. The DPoS protocol, derived from the PoS protocol, involves the voting and electing mechanisms so as to classify participating nodes as different roles to perform different functions in the consensus process. In comparison to the PoW and PoS protocols, the DPoS protocol enables faster block validation [37].

PBFT is a widely used and well-studied consensus algorithm, and the whole consensus process of it consists of five steps: *Request, Pre-prepare, Prepare, Commit, and Reply* [38]. Existing research has demonstrated the effectiveness of the PBFT consensus algorithm for deployment on resource-constrained IoT devices [39]. To further adapt to large-scale IoT networks with a huge number of device nodes, the consensus protocol adopted in this study is a combination of PBFT and DPoS, called the BFT-DPoS consensus protocol, and it inherits the delegate election mechanism in DPoS [24]. In contrast to PBFT, the BFT-DPoS consensus protocol only involves a subset of delegate nodes (i.e., validation nodes) rather than all device nodes during the voting process. For example, there is one client, one primary node, and three other validation nodes. Once the consensus is triggered, the primary node broadcasts a *Pre-prepare* proposal to other validation nodes. During *Prepare* and *Commit* phases, all validation nodes exchange messages to check the reliability and validity of received messages. A node steps into the subsequent phase after receiving more than $\frac{2}{3}$ acknowledgments that include its own.

According to the BFT-DPoS consensus protocol, each general node in the framework can store/fetch transaction data into/from the blockchain system. At the same time, only a part of the nodes can be elected as validation nodes in terms of the number of stakes and available computation resources each node holds. Therefore, when one new block is formed, the new block proposal needs to be broadcasted to other validation nodes. Only if this block has been verified and most of the validation nodes have reached consensus, will it be appended to the main blockchain. Additionally, the communication complexity of PBFT increases exponentially with the number of participating nodes, so there is a trade-off between the communication complexity and security [40].

## 4. System Model

In this section, the system model adopted in this work is introduced. As illustrated in Figure 2, we propose an MEC-enabled blockchain framework for IoT networks, which comprises three parts, i.e., the IoT network including various smart devices, an MEC system including the BS and MEC servers, and a blockchain system based on the DPoS mechanism.

In the IoT network, smart devices, e.g., vehicles, cell phones, security surveillance, etc., collect some ambient data that must be securely stored/processed or shared with other IoT smart devices. As a result, we consider two types of data transactions among smart devices: (1) data storage/processing and (2) data sharing, both of which are stored in the blockchain system for distributed and secure data storage/retrieval. The nodes in the blockchain system are classified into three categories: (1) general nodes (GNs) that consist of all IoT devices, (2) validation nodes (VNs) that are selected out of GNs based on a specific stake distribution according to the DPoS mechanism, and (3) one primary node (PN) that is selected from VNs and authorized to produce blocks at a specific decision epoch. The blockchain system is mainly responsible for the secure storage/retrieval of transaction data from the IoT network. To achieve this goal, the blockchain system must generate blocks and reach consensus, where GNs receive/transmit transaction data from/to other nodes, VNs conduct the blockchain consensus process, and the PN is authorized to generate blocks within a specific time period. Moreover, the MEC system is responsible for sharing the computational pressure in the IoT network and blockchain system to achieve efficient data processing and blockchain consensus. In the following subsections, we will detail the

network model, MEC model, and blockchain model, respectively. The notations used in this paper are summarized in Table 2.

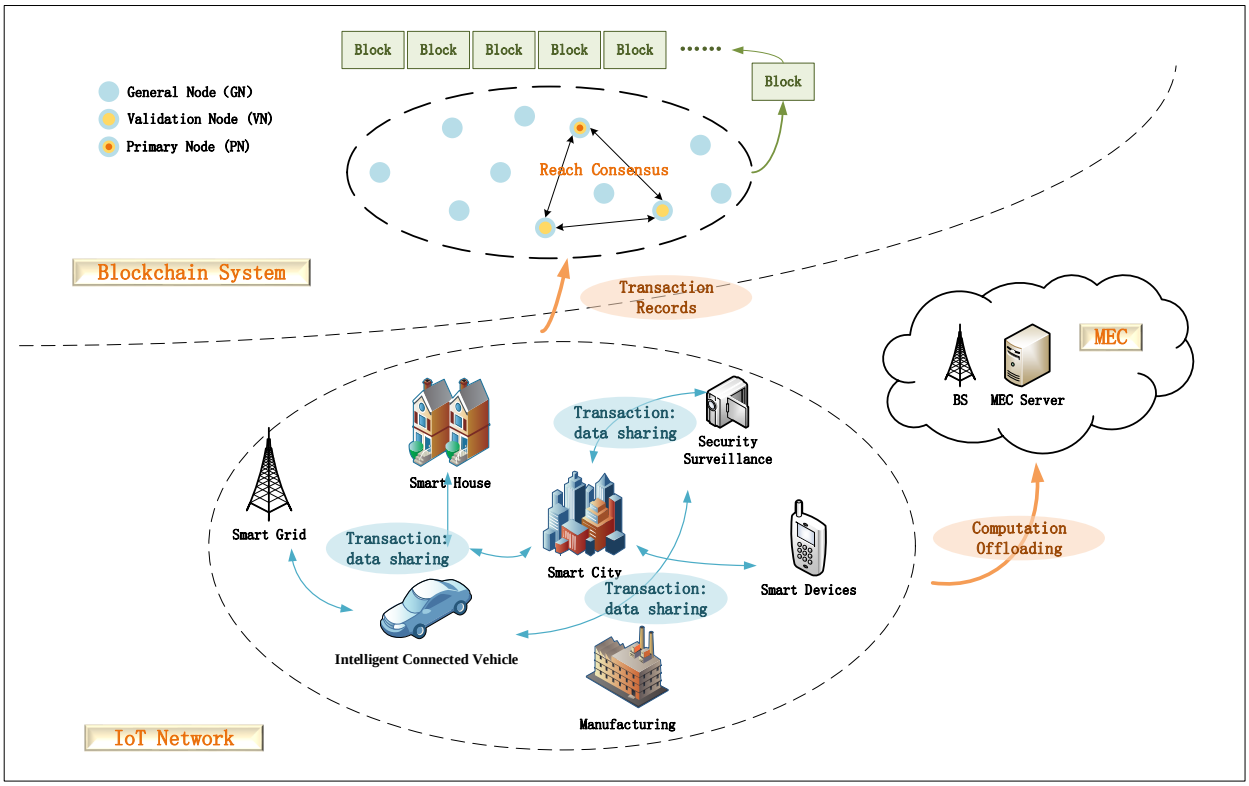


**Figure 2.** MEC-enabled blockchain system in the heterogeneous IoT networks.

**Table 2.** Notation definitions.

| Notation | Description |
|---|---|
| $\Phi_G$ | The set of GNs. |
| $\Phi_V$ | The set of VNs. |
| $M$ | The number of GNs. |
| $K$ | The number of VNs. |
| $L_n$ | The number of time slots included in decision epoch $n$. |
| $\boldsymbol{h}_m(\tau)$ | Channel vector between GN $m$ and the BS at time slot $\tau$. |
| $\boldsymbol{y}(\tau)$ | Received signal of the BS at time slot $\tau$. |
| $\rho_m$ | Normalized temporal channel correlation coefficient of GN $m$. |
| $\boldsymbol{H}(\tau)$ | Channel matrix from all the nodes to the BS at time slot $\tau$. |
| $\gamma_m(\tau)$ | The receiving SINR of GN $m$ at time slot $\tau$. |
| $\phi_m(n)$ | ZF detection vector for GN $m$ at time slot $\tau$. |
| $\lambda_m$ | Task arrival rate of GN $m$. |
| $U_m(n)$ | Queue length of GN $m$'s task buffer at decision epoch $n$. |
| $a_m(n)$ | Number of task arrivals of GN $m$ at decision epoch $n$. |
| $f_m(\tau)$ | CPU frequency scheduled for local execution of GN $m$ at time slot $\tau$. |
| $C_m(F_m)$ | CPU cycles required per one task bit (allowable CPU-cycle frequency) at GN $m$. |
| $p_{o,m}(n)$ | Transmission power of GN $m$ for computation offloading at decision epoch $n$. |
| $d_{o,m}(\tau)(D_{o,m}(n))$ | Data transmitted by GN $m$ for computation offloading at time slot $\tau$ (decision epoch $n$). |
| $p_{l,m}(n)$ | Power consumption of GN $m$ for local execution at decision epoch $n$. |
| $d_{l,m}(\tau)(D_{l,m}(n))$ | Data processed by GN $m$ via local execution at time slot $\tau$ (decision epoch $n$). |
| $P_{o,m}(P_{l,m})$ | Maximum transmission power (local execution power) of GN $m$. |
| $\nu_m(\tau)$ | Computation rate of GN $m$ during time slot $\tau$. |
| $\mathbf{Y}(n)$ | The set of stakes IoT nodes hold at decision epoch $n$. |
| $I_m(n)(I_P(n))$ | Block interval of GN $m$ (PN) at decision epoch $n$. |
| $T_F(n)$ | Latency time to finality at decision epoch $n$. |
| $S^B(n)$ | Block size at decision epoch $n$. |
| $\Xi(n)$ | Blockchain transaction throughput at decision epoch $n$. |

### *4.1. Network Model*

In this paper, we assume that the blockchain system has $M$ GNs denoted by $\Phi_G$ and $K$ VNs. For each IoT node $m \in \Phi_G$, it conducts data storage/processing and data sharing within the IoT network. Meanwhile, it acts as a part of the blockchain system. Specifically, $K$ VNs, denoted by $\Phi_V$, $\Phi_V \subseteq \Phi_G$, are selected out of $\Phi_G$ in terms of particular rules [41]. These VNs are responsible for collecting, validating, and packaging the transactions generated by smart devices into a block. Furthermore, this new block is appended to the blockchain after the PN broadcasts the block proposal to other VNs and a consensus is reached.

As shown in Figure 3, the MEC-enabled blockchain framework is implemented using a discrete-time model in which time is partitioned into multiple decision epochs, and the $n$-th epoch, $n \in \{0, 1, \cdots, N_{max}\}$, has $L_n$ basic time slots which has an identical duration $\tau_0$ and is indexed by $\tau \in \{0, 1, \cdots\}$. Thus, each decision epoch $n$ has a dynamic duration $L_n\tau_0$, where $L_n \in \{1, 2, \cdots, \dot{L_n}\}$ varies for each decision epoch $n$ and is determined by the block interval of PN $I_P(n)$. For each decision epoch $t \in \mathcal{T}$, the wireless channel condition, task arrival, and power allocation policy of each GN are varied. Therefore, aiming to balance data processing efficiency, security, and average energy consumption, each VN needs to determine the block size, block interval, and task offloading policy in each epoch. Assume that the PN produces a new block with block size $S^B(n)$ and block interval $I_P(n)$ in turns within the $n$-th epoch. Specifically, block size $S^B(n)$ indicates the number of bits included in a new block produced by the PN at the end of epoch $n$, while block interval $I_P(n)$ indicates the time required for the VN to generate a new block in each epoch $n$.

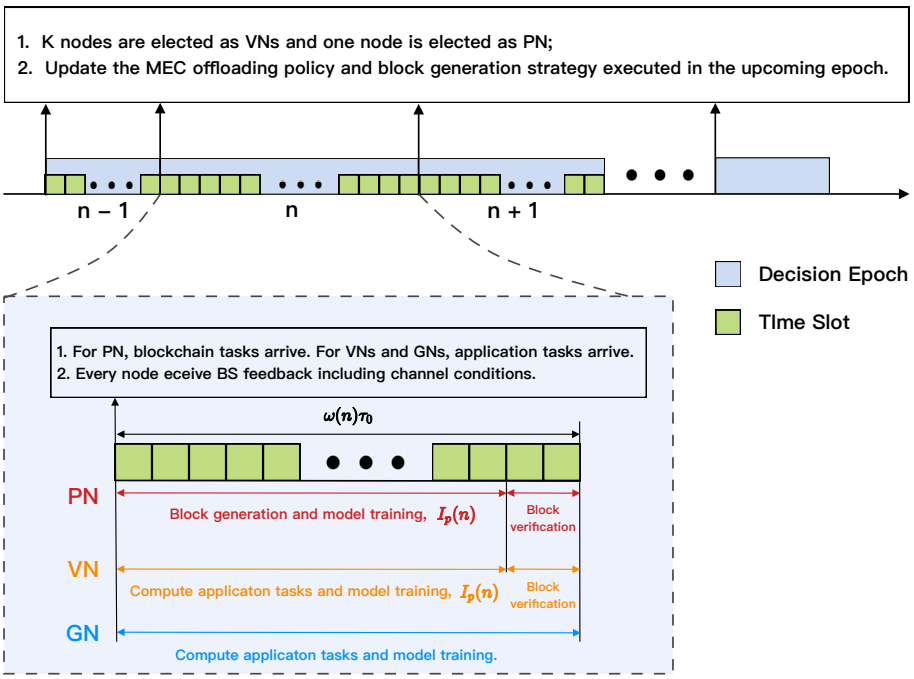


**Figure 3.** The relationship between time slot $\tau$ and decision epoch $n$.

In the proposed framework, we analyze a 5G macro-cell base station (BS) having $N_a$ antennas that handle the uplink communications of numerous IoT nodes with a single antenna using linear zero-forcing (ZF) detection [42], which is simple and efficient [43]. In this study, we assume that the number of antennas at the BS exceeds the number of mobile nodes, i.e., $N_a > M$. For each time slot $\tau$, we denote the channel vector of each GN $m$ as $\boldsymbol{h}_m(\tau) \in \mathbb{C}^{N_a \times 1}$, and therefore the nearest BS's received signal can be represented by:

$$\boldsymbol{y}(\tau) = \sum_{m=1}^{M} \boldsymbol{h}_m(\tau)\sqrt{p_{o,m}(\tau)}s_m(\tau) + \boldsymbol{n}_G(\tau), \quad (1)$$

where $p_{o,m}(\tau) \in [0, \dot{P}_o]$ denotes the uplink power for GN $m$ to offload transaction tasks with the upper bound $\dot{P}_o$, $s_m(\tau)$ denotes the complex data symbol. In addition, $\boldsymbol{n}_G(\tau) \sim \mathcal{CN}_{\dashv}(\mathbf{0}, \sigma_R^2 \boldsymbol{I}_{N_a})$ is a noise vector where $\sigma_R^2$ denotes the variance and $\boldsymbol{I}_{N_a}$ is a $N_a \times N_a$ identity matrix. Furthermore, we adopt the following Markov block fading auto-regressive model [44] to define the temporal relation between decision epochs and movement for each GN $m$:

$$\boldsymbol{h}_m(\tau+1) = \rho_m \boldsymbol{h}_m(\tau) + \sqrt{1-\rho_m^2}\boldsymbol{e}(\tau+1), \tag{2}$$

where $\rho_m = J_0(2\pi f_m^d \tau_0)$ denotes the normalization of correlation function between time slots $\tau+1$ and $\tau$ in terms of Jake's fading spectrum, and the error vector $\boldsymbol{e}(\tau) \sim \mathcal{CN}(\mathbf{0}, \sigma_R^2 \boldsymbol{I}_{N_a})$ is complex Gaussian and independent identically distributed with $\boldsymbol{h}_m(\tau)$. It is worth noting that $f_m^d$ and $J_0(\cdot)$ denote the Doppler frequency of GN $m$ and first-order Bessel function, respectively.

The $N \times M$ channel matrix between the considered 5G macro-cell BS and $M$ GNs is represented by $\boldsymbol{H}(\tau) = \left[\boldsymbol{h}_1(\tau), \boldsymbol{h}_2(\tau), \cdots, \boldsymbol{h}_M(\tau)\right]$. The linear zero-forcing detection is derived by $\boldsymbol{H}^{\dagger}(\tau) = \left(\boldsymbol{H}^H(\tau)\boldsymbol{H}(\tau)\right)^{-1}\boldsymbol{H}^H(\tau)$. After applying the ZF detector, each node's signal to interference-plus-noise ratio (SINR) is calculated by [43]:

$$\gamma_m(\tau) = \frac{p_{o,m}(\tau)}{\sigma_R^2\left[\left(\boldsymbol{H}^H(\tau)\boldsymbol{H}(\tau)\right)^{-1}\right]_{mm}}, \tag{3}$$

where $[\boldsymbol{I}]_{m_1 m_2}$ denotes the $(m_1, m_2)$-th item in matrix $\boldsymbol{I}$.

For each decision epoch $n$, we assume that the channel condition and the power allocation (i.e., $p_{o,m}(n)$ and $p_{l,m}(n)$) is consistent throughout one decision epoch and updated at the first time slot of each epoch.

### *4.2. MEC Model*

In this subsection, we will show how each GN $m$ makes use of an adaptive compute offloading policy to support blockchain-enabled IoT networks. $a_m(n)$ (bit) is the number of computing tasks during the decision epoch $n$, which is assumed to be processed since decision epoch $n+1$. In addition, we consider that $a_m(n)$ is independent and identically distributed throughout different decision epochs and there is an average task-arrival rate $\lambda_m = \mathbb{E}\left[a_m(n)\right]$ based on Poisson distribution. In general, $a_m(n)$ is considered as ordinary application tasks, while for the node elected as PN at decision epoch $n$ we consider $a_P(n)$ as block-generation tasks. Furthermore, the processing of computation tasks is considered fine-grained [45]. Therefore, within the $n$-th decision epoch , $D_{l,m}(n) = L_n d_{l,m}(\tau)$ denotes partial bits of computation tasks that are allocated to be processed locally, while $D_{o,m}(n) = L_n d_{o,m}(\tau)$ represents some other bits that are transferred to and processed by the edge server. We denote $U_m(n)$ as the queue length of the GN $m$'s task buffer at decision epoch $n$'s commencement, and then $U_m(n+1)$ can be expressed by:

$$U_m(n+1) = \Big[U_m(n) - \left(D_{l,m}(n) + D_{o,m}(n)\right)\Big]^+ + a_m(n), \tag{4}$$

where $[x]^+ = max(0, x)$ and $U_m(0) = 0$.

#### 4.2.1. Local Computation

This section demonstrates the number of data bits handled locally in relation to the power allocated for local computing $p_{l,m}(\tau) \in [0, \dot{P}_l]$. By chip voltage adjustment based on the dynamic voltage and frequency scaling technology [46], the central processing unit (CPU) frequency (Hz) scheduled for the time slot $\tau$ is expressed as:

$$f_m(\tau) = \sqrt[3]{p_{l,m}(\tau)/\iota}, \tag{5}$$

in which $\iota$ denotes the effective switching capacitance of the chip, which varies according to its architecture [46]. Additionally, we have $0 \leq f_m(\tau) \leq F_m$ and $F_m = \sqrt[3]{\dot{P}_l/\iota}$, which is the highest permitted CPU frequency of GN $m$ depending on system capability. Consequently, the number of locally processed bits during the time slot $\tau$ is calculated by multiplying the time (s) by the computation rate of the device CPU (bit/s). Specifically, the computation rate of the device CPU (bits/s) is derived by multiplying the device CPU frequency (cycles/s) by the number of task bits that the CPU can process per cycle (bits/cycle). That is,

$$d_{l,m}(\tau) = \tau_0 f_m(\tau) C_m^{-1}, \tag{6}$$

where $C_m$ (cycles/bit) denotes the number of CPU cycles necessary for GN $m$ to compute one data bit and it is measured and determined with offline measurement [47].

#### 4.2.2. Edge Computation

To start with, we assume that the MEC server can handle different computation tasks with a minimal processing delay due to adequate computational resources such as a high-frequency multicore CPU. In addition, as a result of the small-sized computation output, it can be assumed that the feedback delay between BS and node can be ignored. Based on (3) and with the uplink communication power $p_{o,m}(\tau)$, the number of task bits offloaded by GN $m$ within the time slot $\tau$ is calculated by:

$$d_{o,m}(\tau) = \tau_0 W log_2\big(1 + \gamma_m(\tau)\big), \tag{7}$$

where $W$ denotes the system bandwidth.

Therefore, the computation rate (bits/s) of GN $m$ during time slot $\tau$ is calculated by:

$$\nu_m(\tau) = f_m(\tau) C_m^{-1} + W log_2\big(1 + \gamma_m(\tau)\big). \tag{8}$$

According to the MEC computation model, each IoT node with stochastic task arrivals can utilize the nearby MEC server to process the compute-intensive tasks efficiently and adjust the ratio of computation offloading dynamically.

### *4.3. Blockchain System*

In this subsection, we present the considered blockchain system. The blockchain system can provide data security and privacy guarantee to the IoT networks as an overlaid system. Each node inside the blockchain system is capable of collecting data transactions from the IoT networks, while only a few with a large number of stakes are elected as VNs for packaging and validating blocks. We assume that $\mathbf{Y}(n) = \{\mathrm{Y}_1(n), \mathrm{Y}_2(n), \cdots, \mathrm{Y}_M(n)\}$ denotes the set of stakes IoT nodes hold based on the BFT-DPoS consensus algorithm [24] during the $t$-th decision epoch. The stake of each GN is updated when consensus is reached on a new block, and the latest stake distribution is aggregated to the BS. For the start of each decision epoch $n$, the BS distributes the latest stake distribution $\mathbf{Y}(n)$ it has recorded to all GNs.

In the following part, the details of the most significant criteria for evaluating the blockchain system performance are described.

#### 4.3.1. Decentralization

To prevent block packaging and verification power from being monopolized, it is necessary to quantify the degree of decentralization to ensure the long-term fairness of the blockchain system [41]. We make use of *Gini coefficient*, which has been widely used to measure wealth or income inequality [48], evaluate "contrast intensity" [49] and capture "system inequality" [50] in existing works. Focusing on the decentralization of VNs, we

take the stake distribution of VNs into account in this study. Therefore, the Gini coefficient of stake distribution can be derived to characterize the decentralization:

$$G(\mathbf{Y}(n)) = \frac{\sum_{i,j\in\Phi_V, i\neq j}\left|\Upsilon_i(n) - \Upsilon_j(n)\right|}{K\sum_{i\in\Phi_V}\Upsilon_i(n)}. \tag{9}$$

$G(\mathbf{Y}(n))$ is within $[0,1]$ in which the extremes 0, 1 denote the perfect uniformity and maximal inequality among stake values, respectively. Hence, to guarantee that VNs' stake distribution of a blockchain system is decentralized, the Gini coefficient $G(\mathbf{Y}(n))$ should satisfy the constraint:

$$G(\mathbf{Y}(n)) \leq \eta, \tag{10}$$

where we have $\eta \in [0,1]$. For simplicity, we assume that $G(\mathbf{Y}(n)) \leq \eta$ is always satisfied in this work.

#### 4.3.2. Latency Time to Finality and Throughput

The concept of latency time to finality (LTF) is adopted to characterize the latency of the blockchain system, which is the latency that one data transaction record becomes irreversible once it has been committed to the blockchain system [15]. For latency-sensitive applications, it is important to guarantee that the latency is within the user's tolerance. The LTF $T_F(n)$ including the time cost for block validation $T_c(n)$ and generation $I_P(n)$ is expressed as:

$$T_F(n) = T_c(n) + I_P(n), \tag{11}$$

where $T_c(n)$ denotes the consensus latency which includes the time cost for packet transmission $T_{tr}(n)$ and packet verification $T_v(n)$ that includes message authentication codes (MACs) generation, request signature, and MACs verification [51]. Note that the latency required for one consensus process in the simulation of this paper is in the order of milliseconds. The majority of real-world mobile IoT devices (e.g., cell phones, smartwatches, etc.) generally have a small displacement within 10 ms. Therefore, we assume that the primary node and consensus nodes usually can provide stable services in one consensus process.

As introduced in Section II, each decision epoch $n$ has a dynamic duration $L_n\tau_0$. We assume that the duration of decision epoch $n$ is determined by the LTF $T_F(n)$, and thus we have

$$L_n = \left\lceil \frac{T_F(n)}{\tau_0} \right\rceil \tag{12}$$

As in [52], the blockchain transaction throughput of the proposed framework within the $n$-th decision epoch is derived by:

$$\Xi(n) = \frac{\lfloor S^B(n)/\chi \rfloor}{T_F(n)}, \tag{13}$$

where $\chi$ denotes average transaction size. $S^B(n)$ is the block size and derived by:

$$S^B(n) = D_{l,P}(n) + D_{o,P}(n). \tag{14}$$

where $P$ denotes the PN.

#### 4.3.3. Adversarial Fraction

To ensure the blockchain system's security performance, it is vital to prevent transactions from being unilaterally tampered with or reversed. The adversarial fraction of hashing power that an adversary can control without endangering system security is one kind of fundamental performance measurement of a blockchain system [52]. For the PBFT-based consensus protocols, unambiguous finality can be reached under the assumption that less

than a $\frac{1}{3}$ fraction of the nodes are adversarial [24]. Therefore, the following constraint needs to be satisfied:

$$f \leq \left\lfloor \frac{K-1}{3} \right\rfloor, \tag{15}$$

in which $f$ denotes the number of adversarial VNs. For simplicity, in this work, when there are $K$ VNs involved in the consensus process, we assume that $K \geq 3f + 1$. In other words, it is assumed that the above constraint is always met in this work.

## 5. DDPG-Based Performance Optimization Framework

In the following part, we investigate the optimal block generation and computation offloading policies to maximize the transaction throughput of the proposed framework under the constraints of latency and system resources. The joint optimization problem is formulated as an MDP. Moreover, in order to deal with the dynamic and large-dimensional properties of the above-mentioned systems, we develop a DRL-based scheme. Specifically, elements included in the action space (i.e., power allocation and block interval) are continuous variables, which motivates us to employ the DDPG algorithm so as to achieve better performance than the DQN-based approach with a discrete action space [53].

The architecture of the DDPG-based framework is shown in Figure 4 and the DDPG agent is implemented in each GN. To deploy the framework, we first define the problem formulation and construct the state space, action space, and reward function. Then, we design a DDPG-based algorithm to solve it as follows.

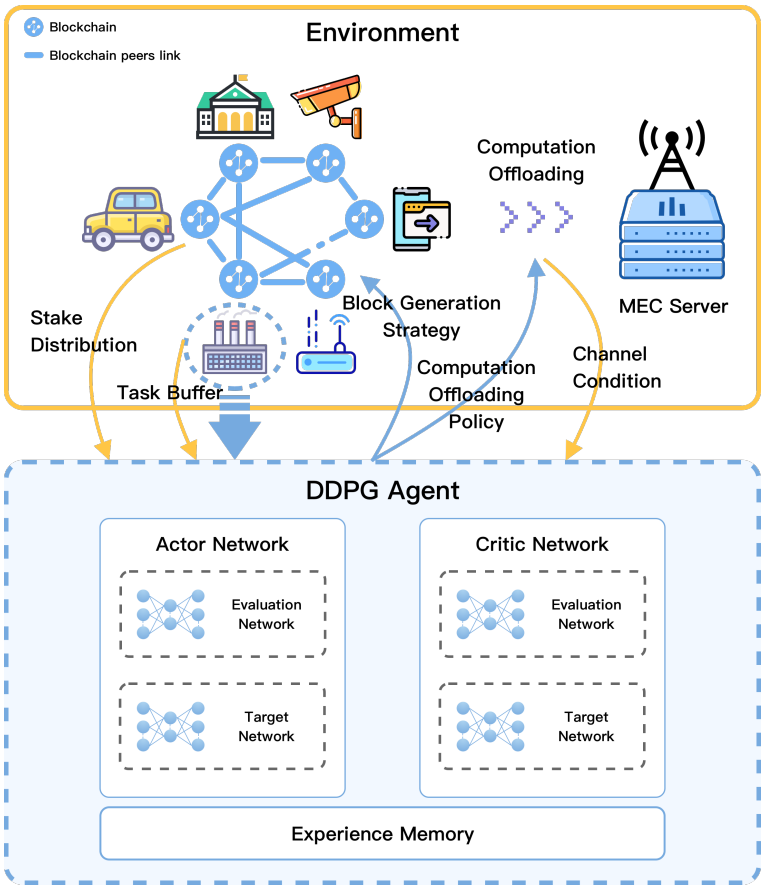


**Figure 4.** The architecture of the DDPG-based framework.

### *5.1. Problem Formulation*

Due to the difficulty of obtaining the state transition probabilities and reward values in advance which are related to user mobility, node workload, etc., the optimization problem is constructed as an MDP. As mentioned above, to learn the resource-aware dynamic block

generation and computation offloading strategies, we propose maximizing the blockchain transactional throughput and node computation rate while guaranteeing the security and decentralization of data storing/processing with the constraints of computation resources and latency. In other words, each GN $m$ needs to solve the following optimization problem in each decision epoch:

$$\begin{aligned} &\mathcal{P}1: \max_{\{p_{l,m}(n),p_{o,m}(n),I_P(n)\}} \mathbb{E}\left[\sum_{n=1}^{T_{max}}\left[\omega\Xi(n)+(1-\omega)\xi\sum_{m=1}^{M}\nu_m(n)\right]\right] \\ &\mathcal{C}1: T_F(n) \le \dot{L_n} \times \tau_0 \\ &\mathcal{C}2: 0 \le p_{o,m}(n) \le \dot{P}_o, 0 \le p_{l,m}(n) \le \dot{P}_l \end{aligned} \tag{16}$$

where $\dot{L_n}$ is the upper bound of $L_n$, $\omega(0 < \omega < 1)$ denotes the weight factor for integrating the two objective components, and $\xi$ is a mapping factor which ensures that the blockchain transactional throughput and the MEC total computation rate are of the same order of magnitude.

The constraints $\mathcal{C}1$ and $\mathcal{C}2$ specify latency and power consumption limits, respectively. Note that, for satisfying the latency requirement of IoT applications, it is assumed that each block should be published and validated within $\dot{L_n}(L_n \ge 1)$ consecutive time slots.

A decentralized dynamic performance optimization strategy will be learned separately at each node, which determines the block interval and power allocation for both local computing and edge computing, depending on the local observation of the environment. Note that the DDPG-based online learning process is totally model-free, which means this algorithm does not require each node to have prior knowledge of the blockchain and MEC systems.

### *5.2. State Space*

It is worth noting that collecting a full observation of the system for all nodes and then distributing them to each node requires high system overheads. Therefore, it is assumed that each node's state is decided by the observation of the system from its own perspective to avoid high system overheads and make the framework more scalable.

For each decision epoch $n$, the workload of each GN's task buffer $U_m(n)$ is updated in terms of (4). Meanwhile, GN receives one message from the BS conveying the stake distribution $\mathbf{Y}(n)$ and the latest SINR of GN to BS $\gamma_m(n-1)$. In addition, $\boldsymbol{h}_m(n)$ for the forthcoming uplink communication is calculated according to (2). As a result, the state space is defined as:

$$s_{m,n} = \left[U_m(n), \mathbf{Y}(n), \phi_m(n-1), \boldsymbol{h}_m(n)\right], \tag{17}$$

in which the projected power ratio after ZF detection $\phi_m(n)$ is calculated by:

$$\begin{aligned} \phi_m(n) &= \frac{\gamma_m(n)\sigma_R^2}{p_{o,m}(n)\|\boldsymbol{h}_m(n)\|^2} \\ &= \frac{1}{\|\boldsymbol{h}_m(n)\|^2\left[\left(\boldsymbol{H}^H(n)\boldsymbol{H}(n)\right)^{-1}\right]_{mm}}. \end{aligned} \tag{18}$$

In addition, ZF detection projects the received signal $\boldsymbol{y}(n)$ into a space orthogonal to the one spanned by channel vectors of other nodes so that GN $m$'s offloaded symbols can be decoded without inter-stream interference [54].

### *5.3. Action Space*

According to the state $s_{m,n}$, each GN $m$ will independently select an action $a_{m,n}$ which includes block interval $I_m(n)$, the allocated power for local computing $p_{l,m}(n)$ and the

allocated power for computation offloading $p_{o,m}(n)$. Consequently, the action space in decision epoch $n$ can be defined as:

$$a_{m,n} = \left[p_{l,m}(n), p_{o,m}(n), I_m(n)\right], \tag{19}$$

where we have $p_{l,m}(n) \in [0, \dot{P}_l]$, $p_{o,m}(n) \in [0, \dot{P}_o]$ and $I_m(n) \in [0, \dot{I}]$. Note that the output action of the DDPG algorithm directly maps the states to the optimal power allocation and block generation policy in a continuous action space, which is different from other conventional DRL algorithms where the output is the probability distribution across a discrete action space. Therefore, the dimensional disaster can be avoided in the DDPG algorithm [53].

### *5.4. Reward Function*

Considering that each node agent's behavior is incentive driven, the reward function is important to the convergence of DDPG algorithm. According to the objective of our joint performance optimization problem defined in (16), we construct the reward function $r_{m,n}$ which GN $m$ receives after decision epoch $n$ as:

$$r_{m,n} = \begin{cases} \omega\Xi(n) + (1-\omega)\xi \sum_{m=1}^{M} \nu_m(n), & \text{if } \mathcal{C}1 - \mathcal{C}2 \text{ are satisfied} \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

Furthermore, note that the value function of node agent $m$ starting from a random initial state $s_{m,1}$ under the policy $\mu_m$ can be expressed by

$$\begin{aligned} V_{\mu_m}(s_{m,1}) &= \mathbb{E}_{\mu_m}\left[\sum_{n=1}^{\infty} \gamma^{n-1} r_{m,n} \middle| s_{m,1}\right] \\ &= \mathbb{E}_{\mu_m}\left[r_{m,n} + \gamma^{n-1} \cdot V^{\mu_m}(s_{m,n+1}) \middle| s_{m,1}\right], \end{aligned} \tag{21}$$

where $\gamma \in [0,1]$ is the discounting factor in the Bellman equation. The value function $V_{\mu_m}$ can be used to quantify the performance of the policy $\mu_m$ via an infinite horizon and discounted MDP [55] at node agent $m$. The following average transactional throughput

$$\bar{O}_m(s_{m,n}) = \mathbb{E}\left[\lim_{T\to\infty} \frac{1}{T} \sum_{i=t}^{T} r_{m,i} \middle| s_{m,n}\right], \tag{22}$$

will be maximized by implementing the optimal block generation and computation offloading policy $\mu_m^*$.

### *5.5. DDPG-Based Algorithm Design*

To solve this problem, we provide a model-free and DRL-based approach for finding the optimal block generation and computation offloading strategies jointly. Moreover, owing to the continuous action space of our MDP model, which is intractable by using traditional learning methods, we suggest a DDPG-based approach to address this problem. The DDPG algorithm is a widely used model-free and off-policy algorithm for continuous action spaces. This subsection first introduces the basic mechanism of the DDPG algorithm and then describes the approach to solve the considered problem.

#### 5.5.1. DDPG Background

As shown in Figure 5, DDPG is a DRL framework that includes two main networks: (1) the actor network and (2) the critic network. Specifically, the actor network is trained for generating the current policy, whereas the critic network is trained for evaluating the advantages and disadvantages of the current policy.

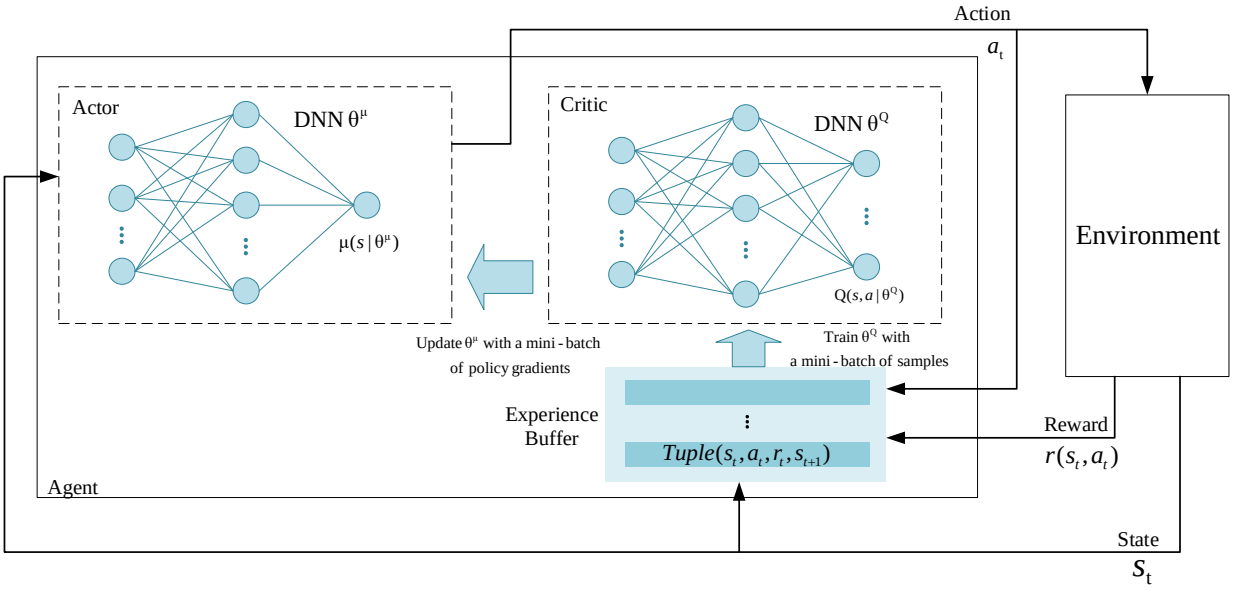


**Figure 5.** The DDPG algorithm: a model-free, off-policy, and actor–critic algorithm.

Specifically, the critic network uses neural networks to simulate real $Q$-table to circumvent the curse of dimensionality. Given the current state $s_n$, the action $a_n$ and the deterministic policy $\mu$, we can write the action-value function as:

$$Q^{\mu}(s_n, a_n) = \mathbb{E}_{s_{n+1}, r_n \sim \Psi}\Big[r(s_n, a_n) + \gamma Q^{\mu}(s_{n+1}, a_{n+1})\Big] \tag{23}$$

where $\Psi$ represents the expectation distribution for $r_n$ and $s_{n+1}$.

Similar to the DQN algorithm [26], the critic function $Q(s_n, a_n|\theta^Q)$ is updated by minimization of the loss function that can be written by:

$$L(\theta^Q) = \mathbb{E}_{s_n \sim \psi^s, a_n \sim \psi^a, r_n \sim \Psi}\Big[\Big(Q(s_n, a_n|\theta^Q) - \epsilon_n\Big)^2\Big] \tag{24}$$

where $\psi^s$ and $\psi^a$ represent the distribution of state $s_n$ and action $a_n$, respectively. $\epsilon_n$ is given by:

$$\epsilon_n = r_n + \gamma Q\big(s_{n+1}, \mu'(s_{n+1})|\theta^Q\big), \tag{25}$$

where $\mu'$ represents the deterministic policy at epoch $n+1$.

The actor function $\mu(s|\theta^{\mu})$ can map a state $s$ to a deterministic action $a$ in a continuous space. Based on the critic function, the policy's updating gradient of the actor is calculated using the chain rule:

$$\nabla_{\theta^{\mu}} J = \mathbb{E}_{s_n \sim \psi^s}\Big[\nabla_a Q(s_n, a|\theta^Q)|_{a=\mu(s_n)} \nabla_{\theta^{\mu}} \mu(s_n|\theta^{\mu})\Big]. \tag{26}$$

Therefore, based on (24) and (26), the actor and critic networks' parameters are softly updated in terms of $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$ and $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$ where $\tau$ is the soft update rate.

#### 5.5.2. The Proposed Algorithm

The detailed DDPG-based optimization algorithm is demonstrated in Algorithm 1 which is implemented in Tensorflow [56]. The algorithm terminates after a preset maximal number of steps $T_{max}$ every episode. For each training episode, the beginning state $s_{m,1}$ is initialized randomly. For each decision epoch $n$, each GN will accumulate and preserve a transition $(s_{m,n}, a_{m,n}, r_{m,n}, s_{m,n+1})$ into its own experience memory $\mathcal{B}_m$. Meanwhile, a

random sample of $Z$ transitions $\{(s_z, a_z, r_z, s_z^{'})\}_{z=1}^{Z}$ from $\mathcal{B}_m$ will be utilized to update the node's own actor and critic networks. After the predefined maximum episodes $K_{max}$, each GN will autonomously learn the resource-aware adaptive block generation and computation offloading policy.

Furthermore, at the testing phase, each node agent will directly load the model learned during the previous training phase, and then interact with the environment, beginning with an empty data buffer $U_m(0)$. Similarly, its current state is determined by local observations of the environment and the corresponding action is selected in terms of the output of the actor network.

**Algorithm 1** DDPG-based Optimization Framework for MEC-enabled Blockchain IoT Systems.

**for** each GN $m \in \Phi_G$ **do**
  **Initialization:** replay memory $\mathcal{B}_m$, critic network $Q(s, a|\theta_m^Q)$, actor network $\mu(s|\theta_m^\mu)$ and corresponding target networks $Q'$ and $\mu'$ with weights $\theta_m^{\mu'} \leftarrow \theta_m^\mu$ and $\theta_m^{Q'} \leftarrow \theta_m^Q$;
**end for**
**for** each episode $\in \{1, 2, \ldots, K_{max}\}$ **do**
  **Initialization:** state $s_{m,1}$ for each GN $m \in \Phi_G$;
  **for** each decision epoch $n = 1, 2, \ldots, T_{max}$ **do**
    **for** each GN $m \in \Phi_G$ **do**
      Select action $a_{m,n} = \mu(s_{m,n}|\theta_m^\mu) + \Delta\mu$ based on the exploration noise $\Delta\mu$ to decide block interval and power allocation;
      Observe reward $r_{m,n}$ and next state $s_{m,n+1}$;
      Store transition data $(s_{m,n}, a_{m,n}, r_{m,n}, s_{m,n+1})$ into replay memory $\mathcal{B}_m$;
      Sample a mini-batch of $Z$ transition tuples $\{(s_z, a_z, r_z, s_z')\}_{z=1}^{Z}$ from memory $\mathcal{B}_m$ at random;
      Update critic network by minimizing the loss $L$:

$$L = \frac{1}{Z}\sum_{z=1}^{Z}\left(Q(s_z, a_z|\theta_m^Q) - \epsilon_z\right)^2;$$

      Update actor policy based on the sampled policy gradient:

$$\nabla_{\theta_m^\mu} J \approx \frac{1}{Z}\sum_{z=1}^{Z} \nabla_a Q(s_z, a|\theta_m^Q)|_{a=\mu(s_z)} \nabla_{\theta_m^\mu}\mu(s_z|\theta_m^\mu);$$

      Update target networks:

$$\theta_m^{\mu'} \leftarrow \zeta\theta_m^\mu + (1-\zeta)\theta_m^{\mu'}$$

$$\theta_m^{Q'} \leftarrow \zeta\theta_m^Q + (1-\zeta)\theta_m^{Q'}$$

    **end for**
  **end for**
**end for**

## 6. Simulation Results and Discussions

We compare the proposed distributed DDPG-based scheme to some other baseline schemes in different simulation scenarios in this part. Additionally, simulation results are provided to demonstrate the proposed DDPG-based framework for performance optimization in the MEC-enabled blockchain IoT system. For the software environment, all code is implemented in Tensorflow Version 1.15.0 with Python 3.7 in a Windows 10 system. The main simulation parameters are summarized in Table 3.

**Table 3.** Simulation parameters.

| Parameter | Value |
|---|---|
| The number of action levels in DQN, $\mathcal{L}$ | 8 |
| The path-loss constant in channel vector, $h_0$ | $-30$ dB |
| The reference distance in channel vector, $d_0$ | 1 m |
| The length of one time slot, $\tau_0$ | 10 ms |
| The pass-loss exponent in channel vector, $\alpha$ | 3 |
| The channel correlation coefficient, $\rho_m$ | 0.95 |
| The Doppler frequency of GN $m$, $f_m^d$ | 70 Hz |
| System bandwidth, $W$ | 1 MHz |
| Average transaction size, $\chi$ | 200 B |
| The maximum transmission power, $\dot{P}_o$ | 4 W |
| The maximum power available for local execution, $\dot{P}_l$ | 4 W |
| The maximum number of time slots for one epoch, $L_n$ | 20 |
| The noise power, $\sigma_R^2$ | $10^{-9}$ W |
| The effective switched capacitance, $\iota$ | $10^{-27}$ |
| The required CPU cycles for each task bit, $C_m$ | 500 cycles/bit |
| The maximum allowable CPU-cycle frequency, $F_m$ | 1.26 GHz |
| The weight factor, $\omega$ | 0.5 |
| The soft update rate for the target networks, $\zeta$ | 0.001 |
| The number of episodes, $K_{max}$ | 1000 |
| The maximum steps of each episode, $T_{max}$ | 1000 |
| The task arrival rate, $\lambda_m$ | 2.5 Mbps |
| The thresholds of decentralization, $\eta_s, \eta_l$ | 0.2, 0.3 |

### *6.1. Simulation Setup*

At the start of each episode, each GN m's channel vector can be predefined as $\boldsymbol{h}_m(0) \sim \mathcal{CN}(0, h_0(d_0/d_m)^\alpha \boldsymbol{I}_{N_a})$. Specifically, $h_0$, $d_0$, $\alpha$, and $d_m$ denote path-loss constant, reference distance, path-loss exponent, and distance between GN $m$ and BS, respectively. During the following decision epochs, $\boldsymbol{h}_m(n)$ is updated in terms of (2) [44].

For the implementation of the DDPG algorithm, we have an actor network learning the policy network approximation $\mu(s|\theta_m^\mu)$ and a critic network predicting the Q-function network approximation $Q(s, a|\theta_m^Q)$ concurrently. Specifically, the DDPG algorithm is constructed with a fully connected neural network that includes one input layer, two hidden layers, and one output layer. In our simulations, we assume that there are 400 and 300 hidden neurons in these 2 hidden layers, respectively. In addition, the activation function for the two hidden layers is chosen to be $f(x) = max(0, x)$, and a sigmoid layer is used to bound the output actions. The discounting factor $\gamma = 0.99$ is used. Moreover, we utilize the adaptive moment estimation (Adam) method to learn the neural network parameters, and meanwhile, the learning rates for the actor and critic networks are set as shown in Table 3 [57]. Similar to [53], the final output layer weights and bias of both the actor network and critic network are initialized from a uniform distribution $[-0.003, 0.003]$ and $[-0.0003, 0.0003]$, while the layer weights of other layers are based on the fan-in of the layer. We adopt the Ornstein–Uhlenbeck process ($\theta = 0.15$, $\sigma = 0.15$) to introduce the temporally correlated noise to explore the action space more efficiently [58].

For comparison, three baseline schemes are introduced in the simulation section:

(1) *Greedy local-execution-first scheme (GD-Local):* For each decision epoch $n$, each node agent $m$ attempts to execute buffered data bits locally first, and then offloads the remainder of computation tasks to the MEC server.
(2) *Greedy computation-offloading-first scheme (GD-Offload):* For each decision epoch $n$, the node agent $m$ attempts to offload buffered task bits to the MEC server first, and then the remaining computation tasks are processed locally.

(3) *DQN-based dynamic offloading scheme (DQN):* DQN is a DRL algorithm with a discrete action space [26]. Specifically, for each node agent $m$, we tune the power allocated for local computing and computation offloading from limited candidate sets $\mathcal{P}_{l,m} = \{0, \frac{\dot{P}_l}{\mathcal{L}-1}, \cdots, \dot{P}_l\}$ and $\mathcal{P}_{o,m} = \{0, \frac{\dot{P}_o}{\mathcal{L}-1}, \cdots, \dot{P}_o\}$, respectively, in which $\mathcal{L}$ denotes the total number of discrete levels.

### 6.2. Performance of the Proposed Scheme

As seen in Figure 6, the convergence performance of the proposed DDPG-based performance optimization scheme and three baseline schemes is presented, respectively. In Figure 6, as the number of episodes increases, the average reward increases as well. The average reward becomes stable after about 800 episodes, which verifies that effective performance optimization strategies can be learned without prior knowledge. In addition, the DDPG-based scheme can achieve about 92.86% of the optimal performance after training 100 episodes. Guo, F. et al. [32] showed that the double-dueling DQN-based scheme required 60 episodes of training to achieve 90.48% of the optimal performance. Feng, J. et al. [59] shows that the A3C-based scheme achieves about 91.86% of the optimal performance after training 150 episodes. Therefore, the DDPG-based scheme does not lead to more system overhead than other schemes.

For Figure 6, it is worth noting that DRL-based schemes outperform both greedy schemes as they cannot dynamically and adaptively allocate computing resources, which verifies the superiority of DRL-based schemes. Even though the DQN-based scheme enables the performance of MEC and blockchain systems in IoT networks to be optimized concurrently, the policy learned by the DQN-based scheme may not be the global optimum as the spaces of power allocation and block interval are continuous. However, the DDPG-based scheme with a continuous action space enables discovering the globally optimum policy. It can be found that the performance of the strategy derived from the DDPG-based scheme is superior to the other three baselines, demonstrating the benefit of our suggested framework for continuous control problems.

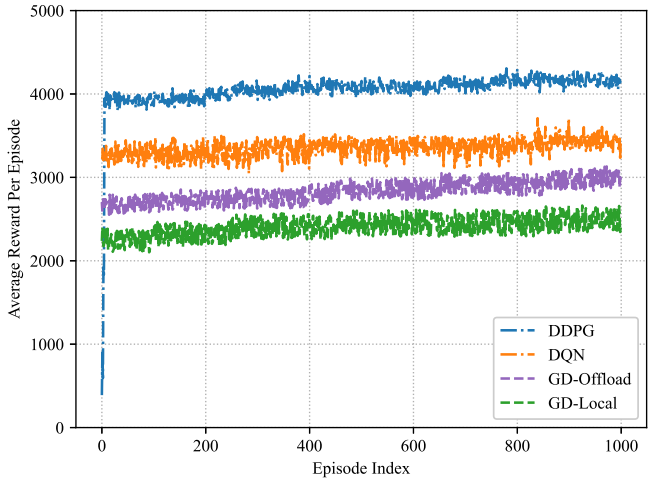


**Figure 6.** Convergence performance based on different schemes in the training process.

In the training process, the parameters of the critic and actor networks are updated continuously. After $K_{max} = 1000$ episodes, we can obtain the trained $c$ for the dynamic block generation and computation offloading in the testing process. Considering a set of task arrival rates $\lambda_m = 1.5 \sim 4.0$ Mbps, we train these four considered schemes independently with the same system parameters for different task arrival rates. As shown in Figures 7–9, as the task arrival rate increases, all schemes achieve a greater average reward, implying that a larger computation demand requires a larger computation cost. Meanwhile, due to the growing computation cost, the average power consumption and buffering delay increase progressively. Furthermore, the suggested DDPG-based scheme can consistently

achieve better average rewards than other schemes, which implies that higher blockchain system throughput and MEC computational efficiency can be obtained under the DDPG-based scheme. Moreover, from Figure 9, it is noted that the DDPG-based scheme slightly compromises the average block interval to obtain the lowest power consumption.

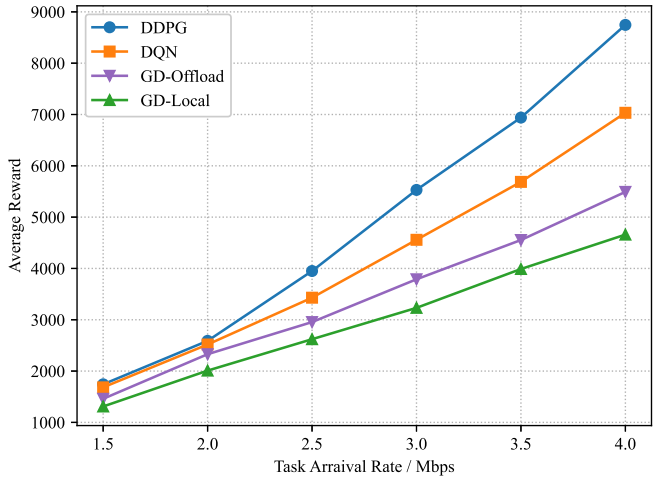


**Figure 7.** Average reward versus task arrival rate.

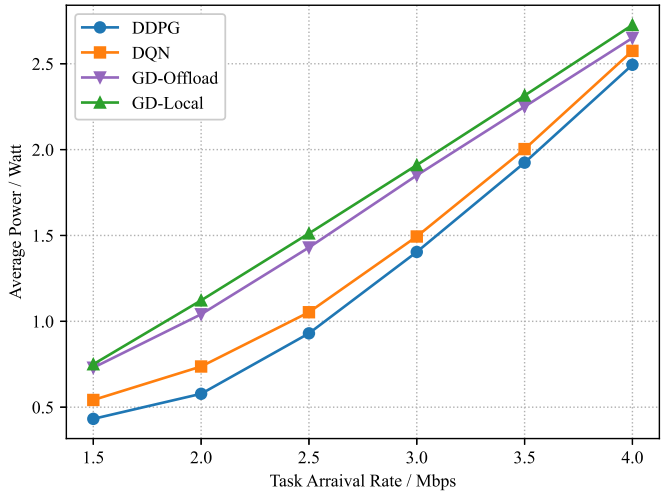


**Figure 8.** Average power consumption versus task arrival rate.

Furthermore, we analyze the testing results of the MEC-enabled blockchain system by setting different threshold values of LTF or local power consumption. Specifically, the effect of the block interval limit is depicted in Figure 10. It has been shown that the average reward increases gradually when the threshold of LTF and task arrival rate grow. An explanation is that each node can process more data transactions over one block under a more relaxed latency constraint. However, at a fixed task arrival rate, the long-term reward value increases and reaches a stable state as the threshold of LTF increases, because the allocable time gradually becomes abundant. As the task arrival rate gets larger, the time to reach saturation is gradually delayed. Moreover, we can see that the DDPG-based scheme can consistently achieve a much higher average reward than the greedy and DQN-based schemes. Figure 11 examines the average reward with different thresholds of local power consumption $\dot{p}_m = \dot{p}_{o,m} + \dot{p}_{l,m}$. Additionally, one observation is that the MEC-enabled blockchain system can process more data transactions when the threshold of the local power consumption increases. However, the average reward does not grow infinitely in Figure 11, because the latency constraint and task arrival rate restrict the maximum amount

of data transactions in one block. These numerical results can help with the design of the MEC-enabled blockchain systems in real-world situations.

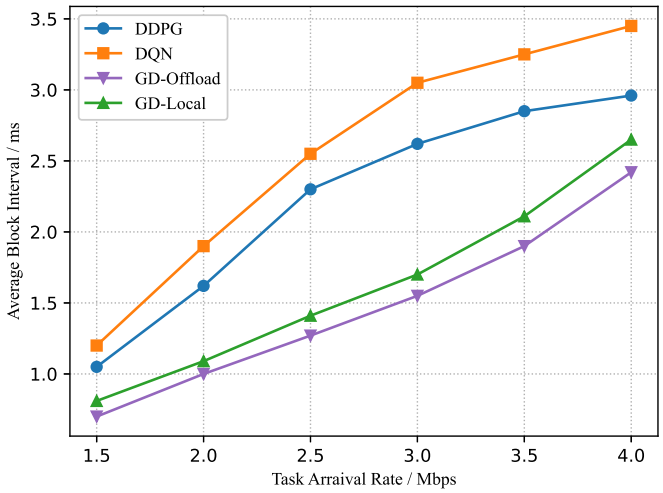


**Figure 9.** Average block interval versus task arrival rate.

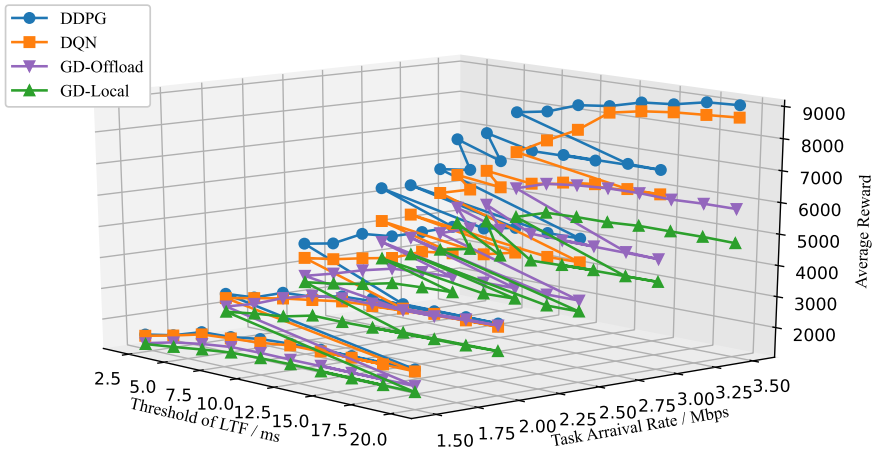


**Figure 10.** Average reward versus threshold of LTF and task arrival rate.

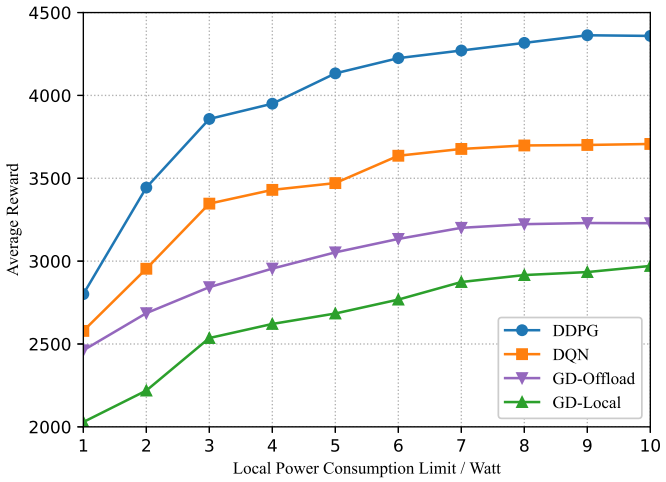


**Figure 11.** Average reward versus threshold of local power consumption.

Figure 12 illustrates the variation of the average reward as the number of nodes increases, demonstrating the scalability of the proposed framework and its ability to adapt to dynamic changes in the user profile. As the number of participating nodes grows, the number of data transactions that need to be packaged increases, and thus the average reward gradually increases. Furthermore, as the number of mobile users increases, the superiority of the DDPG-based scheme over other schemes grows and the performance GD-local scheme declines after the number exceeds 20. Therefore, as long as the size of the computational task is within the processing power of the MEC server, the size of the block generated and validated after one epoch also becomes larger, and meanwhile, the LTF constraint can be satisfied. Based on the proposed joint optimization scheme of local execution power, computation offloading power, block size, and block interval, the DDPG-based algorithm can achieve a better performance than the other considered schemes.

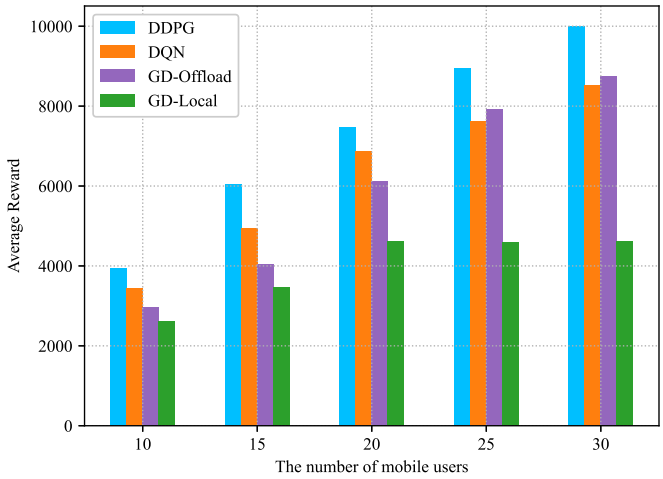


**Figure 12.** Average reward versus the number of mobile users.

Figure 13 shows the effect of different weight factors on the average reward. We can observe that the overall average reward slightly increases as the weight factor $\omega$ decreases. That is because the overall performance of the considered framework mainly depends on the MEC computation rate, and a smaller weight factor indicates a greater emphasis on MEC optimization. The simulation performance of four schemes under different parameters is summarized in Table 4.

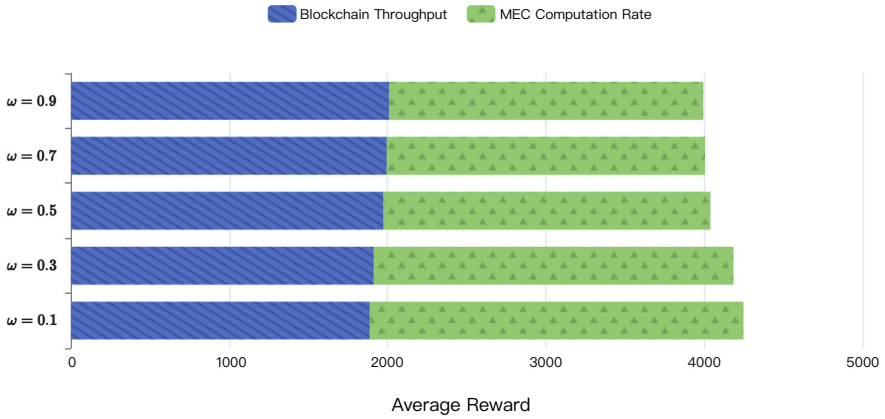


**Figure 13.** The effect of different weight factors on average reward.

**Table 4.** Performance comparison in terms of average reward.

| | | DDPG | DQN | GD-Offload | GD-Local |
|---|---|---|---|---|---|
| *Traning* episodes | *100* | 3856.02 | 3381.07 | 2784.23 | 2371.89 |
| | *500* | 4156.14 | 3414.74 | 2885.67 | 2357.42 |
| | *1000* | 4177.97 | 3445.40 | 3012.81 | 2519.65 |
| *Task arrival rate / Mbps* | *1.5* | 1742.25 | 1683.47 | 1460.23 | 1310.20 |
| | *2.5* | 3950.14 | 3430.28 | 2955.37 | 2621.09 |
| | *3.5* | 6940.87 | 5687.55 | 4554.54 | 3989.32 |
| *Threshold of LTF / ms* | *2.0* | 3761.19 | 3289.24 | 2691.85 | 2478.72 |
| | *7.5* | 4443.35 | 3644.41 | 3250.13 | 2799.44 |
| | *12.5* | 4857.52 | 4228.17 | 3569.60 | 3070.02 |
| *Local power consumption limit / watt* | *2* | 3444.58 | 2954.47 | 2686.43 | 2219.36 |
| | *5* | 4133.25 | 3471.36 | 3053.31 | 2684.68 |
| | *8* | 4317.63 | 3698.01 | 3223.30 | 2916.34 |
| *The number of mobile users* | *10* | 3950.56 | 3430.09 | 2955.40 | 2621.34 |
| | *20* | 7461.89 | 6862.15 | 6110.04 | 4630.71 |
| | *30* | 10,006.53 | 8553.49 | 8742.11 | 4621.98 |

## 7. Conclusions

In this paper, we studied an MEC-enabled blockchain system for future wireless IoT networks and investigated the joint performance optimization problem of blockchain transaction throughput and MEC computational efficiency. The joint problem of MEC computation offloading and block generation policy was formulated, where the power allocation and block interval were optimized. As a result of the time-varying properties of wireless channels in this system, we modeled the joint optimization problem as an MDP. A DDPG-based algorithm was proposed to solve the MDP problem, which can cope with the problem under continuous action space and learn the optimal strategy without prior knowledge of the environment.

The simulation results have demonstrated the proposed scheme outperforms DQN-based and some other greedy (non-joint optimization) schemes under different task arrival rates, thresholds of LTF, and weight factors. The joint optimization scheme can achieve better performance than other schemes with a high convergence rate. Meanwhile, we evaluated its performance in terms of power consumption and block interval, and the simulation results have shown that the joint optimization scheme slightly compromises the average block interval to obtain the lowest power consumption. Additionally, we discussed the impact of the number of mobile users on the convergence performance and the joint optimization scheme always has an advantage over other schemes.

In conclusion, the proposed scheme paves the road for the efficient deployment of blockchain technology in IoT networks, which can be applied in latency-sensitive and security-sensitive IoT applications (e.g., internet of vehicles, virtual reality, smart healthcare, etc.).

**Author Contributions:** Conceptualization, Z.H. and H.G.; software, Z.H.; investigation, Z.H., H.G., T.W., and D.H.; writing—original draft preparation, Z.H. and H.G.; writing—review and editing, Z.H., H.G., T.W., and D.H.; supervision, H.G., T.W., and Y.L.; project administration, H.G. and Y.L.; funding acquisition, H.G. and Y.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially supported by the National Key R&D Program of China under Grant 2019YFB2102404 and 2019YFB2102403. The work of T.W. was supported by the Natural Science Fund of Guangdong Province under Grant 2020A1515010708 and the Natural Science Fund of Shenzhen under Grant JCYJ20210324094609027.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| BS | Base Station |
| CPU | Central Processing Unit |
| DDPG | Deep Deterministic Policy Gradient |
| DPoS | Delegated Proof-of-Stake |
| DQN | Deep Q Network |
| DRL | Deep Reinforcement Learning |
| GN | General Node |
| IoT | Internet of Things |
| LTF | Latency Time to Finality |
| MAC | Message Authentication Code |
| MDP | Markov Decision Process |
| MEC | Mobile Edge Computing |
| PBFT | Practical Byzantine Fault Tolerance |
| PN | Primary Node |
| PoS | Proof of Stake |
| PoW | Proof of Work |
| SINR | Signal to Interference-plus-Noise Ratio |
| VN | Validation Node |
| ZF | Zero Forcing |

## References


1. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for smart cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]
2. Kang, J.; Yu, R.; Huang, X.; Wu, M.; Maharjan, S.; Xie, S.; Zhang, Y. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet Things J.* **2019**, *6*, 4660–4670. [CrossRef]
3. Liu, H.; Zhang, Y.; Yang, T. Blockchain enabled security in electric vehicles cloud and edge computing. *IEEE Netw. Mag.* **2018**, *32*, 78–83. [CrossRef]
4. Wang, H.; Osen, O. L.; Li, G.; Li, W.; Dai, H.; Zeng, W. Big data and industrial Internet of Things for the maritime industry in northwestern Norway. In Proceedings of the TENCON 2015—2015 IEEE Region 10 Conference, Macau, China, 1–4 November 2015; pp. 1–5. [CrossRef]
5. He, J.; Wei, J.; Chen, K.; Tang, Z.; Zhou, Y.; Zhang, Y. Multitier fog computing with large-scale IoT data analytics for smart cities. *IEEE Internet Things J.* **2018**, *5*, 677–686. [CrossRef]
6. Kshetri, N. Can blockchain strengthen the Internet of Things?. *IT Prof.* **2017**, *19*, 68–72. [CrossRef]
7. Pereira, F.; Crocker, P.; Leithardt, V. R. PADRES: Tool for PrivAcy, Data REgulation and Security. *SoftwareX* **2022**, *17*, 100895. [CrossRef]
8. Miller, D. Blockchain and the Internet of Things in the industrial sector. *IT Prof.* **2018**, *20*, 15–18. [CrossRef]
9. Aitzhan, N.Z.; Svetinovic, D. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 840–852. [CrossRef]
10. Li, Z.; Kang, J.; Yu, R.; Ye, D.; Deng, Q.; Zhang, Y. Consortium blockchain for secure energy trading in industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3690–3700. [CrossRef]
11. Liu, Y.; Wang, K.; Lin, Y.; Xu, W. LightChain: A Lightweight Blockchain System for Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3571–3581. [CrossRef]
12. Cong, L.W.; He, Z. Blockchain disruption and smart contracts. *Rev. Financ. Stud.* **2019**, *32*, 1754–1797. [CrossRef]
13. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 557–564. [CrossRef]
14. Chen, W.; Ma, M.; Ye, Y.; Zheng, Z.; Zhou, Y. IoT service based on joint cloud blockchain: The case study of smart traveling. In Proceedings of the 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE), Bamberg, Germany, 26–29 March 2018; pp. 216–221. [CrossRef]

15. Liu, M.; Yu, F.R.; Teng, Y.; Leung, V.C.M.; Song, M. Performance optimization for blockchain-enabled industrial Internet of Things (IIoT) systems: A deep reinforcement learning approach. *IEEE Trans. Ind. Inf.* **2019**, *15*, 3559–3570. [CrossRef]
16. Kang, J.; Xiong, Z.; Niyato, D.; Ye, D.; Kim D.I.; Zhao, J. Toward Secure Blockchain-Enabled Internet of Vehicles: Optimizing Consensus Management Using Reputation and Contract Theory. *IEEE Trans. Veh. Technol.* **2019**, *68*, 2906–2920. [CrossRef]
17. Fan, K.; Wang, S.; Ren, Y.; Yang, K.; Yan, Z.; Li, H.; Yang Y. Blockchain-Based Secure Time Protection Scheme in IoT. *IEEE Internet Things J.* **2019**, *6*, 4671–4679. [CrossRef]
18. Li, W.; Su, Z.; Li, R.; Zhang, K.; Wang, Y. Blockchain-Based Data Security for Artificial Intelligence Applications in 6G Networks. *IEEE Netw.* **2020**, *34*, 31–37. [CrossRef]
19. Satyanarayanan, M. The emergence of edge computing. *Computer* **2017**, *50*, 30–39. [CrossRef]
20. Li, J.; Gao, H.; Lv, T.; Lu, Y. Deep reinforcement learning based computation offloading and resource allocation for MEC. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6. [CrossRef]
21. Zhao, N.; Wu, H.; Chen, Y. Coalition game-based computation resource allocation for wireless blockchain networks. *IEEE Internet Things J.* **2019**, *6*, 8507–8518. [CrossRef]
22. Nguyen, D.C.; Pathirana, P.N.; Ding, M.; Seneviratne, A. Secure computation offloading in blockchain based IoT works with deep reinforcement learning. *arXiv* **2019**, arXiv:1908.07466.
23. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
24. EOS. IO, EOS.IO Technical White Paper v2. Available online: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhite-Paper.md (accessed on 16 March 2018).
25. Watkins, C.J.; Dayan P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
26. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef]
27. Sadawi, A.A.; Hassan, M.S.; Ndiaye, M. A Survey on the Integration of Blockchain With IoT to Enhance Performance and Eliminate Challenges. *IEEE Access* **2021**, *9*, 54478–54497. [CrossRef]
28. Yang, Q.; Wang, H. Privacy-Preserving Transactive Energy Management for IoT-Aided Smart Homes via Blockchain. *IEEE Internet Things J.* **2021**, *8*, 11463–11475. [CrossRef]
29. Nguyen, L.D.; Leyva-Mayorga, I.; Lewis, A.N.; Popovski, P. Modeling and Analysis of Data Trading on Blockchain-Based Market in IoT Networks. *IEEE Internet Things J.* **2021**, *8*, 6487–6497. [CrossRef]
30. Cao ,B.; Liu, W.; Peng, M. Integration of MEC and Blockchain. In *Wireless Blockchain: Principles, Technologies and Applications IEEE*; Wiley: Hoboken, NJ, USA, 2022; pp. 161–177. [CrossRef]
31. Qiu, X.; Liu, L.; Chen, W.; Hong, Z.; Zheng, Z. Online Deep Reinforcement Learning for Computation Offloading in Blockchain-Empowered Mobile Edge Computing. *IEEE Trans. Veh. Technol.* **2019**, *68*, 8050–8062. [CrossRef]
32. Guo, F.; Yu, F.R.; Zhang, H.; Ji, H; Liu, M.; Leung, V.C.M. Adaptive Resource Allocation in Future Wireless Networks With Blockchain and Mobile Edge Computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 1689–1703. [CrossRef]
33. Chen, J.; Wang, W.; Zhou, Y.; Ahmed, S.H.; Wei, W. Exploiting 5G and Blockchain for Medical Applications of Drones. *IEEE Netw.* **2021**, *35*, 30–36. [CrossRef]
34. Lv, Z.; Qiao, L.; Hossain, M.S.; Choi, B. J. Analysis of Using Blockchain to Protect the Privacy of Drone Big Data. *IEEE Netw.* **2021**, *35*, 44–49. [CrossRef]
35. Wang, H.; Wang, Q.; He, D.; Li, Q.; Liu, Z. BBARS: Blockchain-Based Anonymous Rewarding Scheme for V2G Networks. *IEEE Internet Things J.* **2019**, *6*, 3676–3687. [CrossRef]
36. Gu, W.; Li, J.; Tang, Z. A Survey on Consensus Mechanisms for Blockchain Technology. In Proceedings of the 2021 International Conference on Artificial Intelligence, Big Data and Algorithms, Xi'an, China, 28–30 May 2021; pp. 46–49. [CrossRef]
37. Mingxiao, D.; Xiaofeng, M.; Zhe, Z.; Xiangwei, W.; Qijun, C. A review on consensus algorithm of blockchain. In Proceedings of the 2017 IEEE international conference on systems, man, and cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 2567–2572. [CrossRef]
38. Castro, M.; Liskov, B. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.* **2002**, *20*, 398–461. [CrossRef]
39. Meshcheryakov,Y.; Melman, A.; Evsutin, O.; Morozov, V.; Koucheryavy, Y. On Performance of PBFT Blockchain Consensus Algorithm for IoT-Applications With Constrained Devices. *IEEE Access* **2021**, *9*, 80559–80570. [CrossRef]
40. Li, W.; Feng, C.; Zhang, L.; Xu, H.; Cao, B.; Imran, M. A. A Scalable Multi-Layer PBFT Consensus for Blockchain. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1146–1160. [CrossRef]
41. Yang, F.; Zhou, W.; Wu, Q.; Long, R.; Xiong, N.N.; Zhou, M. Delegated Proof of Stake With Downgrade: A Secure and Efficient Blockchain Consensus Algorithm With Downgrade Mechanism. *IEEE Access* **2019**, *7*, 118541–118555. [CrossRef]
42. Yoo, T.; Goldsmith, A. On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 528–541. [CrossRef]
43. Ngo, H.Q.; Larsson, E.G.; Marzetta, T.L. Energy and spectral efficiency of very large multiuser mimo systems. *IEEE Trans. Wirel. Commun.* **2013**, *61*, 1436–1449. [CrossRef]
44. Suraweera, H.A.; Tsiftsis, T.A.; Karagiannidis, G.K.; Nallanathan, A. Effect of feedback delay on amplify-and-forward relay networks with beamforming. *IEEE Trans. Veh. Technol.* **2011**, *60*, 1265–1271. [CrossRef]

45. Kwak, J.; Kim, Y.; Lee, J.; Chong, S. Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems. *IEEE J. Sel. Areas Commun.* **2015**, *33*, 2510–2523. [CrossRef]
46. Burd, T.D.; Brodersen, R.W. Processor design for portable systems. *J. VLSI Signal Process. Syst. Signal Image Video Technol.* **1996**, *13*, 203–221. [CrossRef]
47. Miettinen, A.P.; Nurminen, J.K. Energy efficiency of mobile clients in cloud computing. *HotCloud* **2010**, *10*. [CrossRef]
48. Gini, C. Variability and mutability. *J. R. Stat. Soc.* **1913**, *76*, 619–622.
49. Lin, Z.; Wen, F.; Ding, Y.; Xue, Y. Data-driven coherency identification for generators based on spectral clustering. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1275–1285. [CrossRef]
50. Dai, L.; Jia, Y.; Liang, L.; Chang, Z. Metric and control of system fairness in heterogeneous networks. In Proceedings of the 23rd Asia-Pacific Conference on Communications, Perth, Australia, 11–13 December 2017; pp. 1–5. [CrossRef]
51. Clement, A.; Wong, E.; Alvisi, L.; Dahlin. M. Making Byzantine fault tolerant systems tolerate Byzantine faults. In Proceedings of the Usenix Symposium on Networked Systems Design and Implementation, Boston, MA, USA, 22–24 April 2009; pp. 153–168.
52. Bagaria, V.; Kannan, S.; Tse, D.; Fanti, G.; Viswanath, P. Deconstructing the Blockchain to Approach Physical Limits. *arXiv* **2019**, arXiv:1810.08092.
53. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016. [CrossRef]
54. Tse, D.; Viswanath, P. *Fundamentals of Wireless Communication*; Cambridge University Press: Cambridge, MA, USA, 2005. [CrossRef]
55. Adelman, D.; Mersereau, A.J. Relaxations of weakly coupled stochastic dynamic programs. *Oper. Res.* **2008**, *56*, 712–727. [CrossRef]
56. Martın, A.; Ashish, A. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Available online: https://www.tensorflow.org/ (accessed on 9 November 2015).
57. Kingma, D.P.; Adam, J.B. A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
58. Uhlenbeck, G.E.; Ornstein, L.S. On the theory of the brownian motion. *Phys. Rev.* **1930**, *36*, 823. [CrossRef]
59. Feng, J.; Yu, F.R.; Pei, Q.; Chu, X.; Du, J.; Zhu, L. Cooperative Computation Offloading and Resource Allocation for Blockchain-Enabled Mobile-Edge Computing: A Deep Reinforcement Learning Approach. *IEEE Internet Things J.* **2020**, *7*, 6214–622. [CrossRef]