SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

June 2021, Vol. 64 162303:1–162303:16
https://doi.org/10.1007/s11432-020-3125-y

Intelligent resource allocation in mobile blockchain for privacy and security transactions: a deep reinforcement learning based approach

Zhaolong NING[1,2,3], Shouming SUN[2], Xiaojie WANG[1*], Lei GUO[1], Guoyin WANG[4], Xinbo GAO[5] & Ricky Y. K. KWOK[6]

[1]*School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing* 400065*, China;*
[2]*School of Software, Dalian University of Technology, Dalian* 116620*, China;*
[3]*State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou* 310007*, China;*
[4]*Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing* 400065*, China;*
[5]*Chongqing Key Laboratory of Image Cognition, Chongqing University of Posts and Telecommunications, Chongqing* 400065*, China;*
[6]*Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong* 999077*, China*

Received 1 May 2020/Revised 31 July 2020/Accepted 19 November 2020/Published online 25 April 2021

Abstract In order to protect the privacy and data security of mobile devices during the transactions in the industrial Internet of Things (IIoT), we propose a mobile edge computing (MEC)-based mobile blockchain framework by considering the limited bandwidth and computing power of small base stations (SBSs). First, we formulate a joint bandwidth and computing resource allocation problem to maximize the long-term utility of all mobile devices, and take into account the mobility of devices as well as the blockchain throughput. We decompose the formulated problem into two subproblems to decrease the dimension of action space. Then, we propose a deep reinforcement learning additional particle swarm optimization (DRPO) algorithm to solve the two subproblems, in which a particle swarm optimization algorithm is leveraged to avoid the unnecessary search of a deep deterministic policy gradient approach. Simulation results demonstrate the effectiveness of our method from various aspects.

Keywords mobile blockchain, deep reinforcement learning, mobile edge computing, power allocation, bandwidth allocation

Citation Ning Z L, Sun S M, Wang X J, et al. Intelligent resource allocation in mobile blockchain for privacy and security transactions: a deep reinforcement learning based approach. Sci China Inf Sci, 2021, 64(6): 162303, https://doi.org/10.1007/s11432-020-3125-y

1 Introduction

The advancements of the industrial Internet of Things (IIoT) make it receive great attention in recent years and have been utilized in various fields, e.g., retailing, manufacturing, industrial monitoring, smart transportation, and so on [1–3]. It is inevitable to make transactions between smart devices in IIoT owing to the success of e-commerce [4]. In order to solve the disadvantages of existing traditional centralized market, e.g., privacy and security issues [5,6] as well as a single point of failure [7], a distributed ledger, i.e., blockchain, is proposed. Since it has the characteristics of trustness, decentralization, transparency and tamper-resistant, blockchain enables secure and privacy protected transactions between two untrusted nodes without a third party, which has been applied in many fields, e.g., finance, healthcare and Internet of Things (IoT) [8]. However, in blockchain, each node commonly needs to solve a complex problem puzzle, i.e., proof-of-work (PoW) to complete the priority for appending blocks to get rewards, which is impractical in mobile devices owing to their limited computing power and energy. Then, some researchers turn their eyes to the mobile edge computing (MEC) technique [9, 10].

* Corresponding author (email: xiaojie.kara.wang@ieee.org)

MEC technique can make up the high delay defect of remote cloud when processing tasks for end devices by pulling computing capacity from remote cloud to end devices. Based on blockchain and MEC techniques, mobile blockchain is proposed; i.e., blockchain application is deployed in mobile devices and high computational intense mining puzzle task of each device is offloaded to its nearby MEC servers. With the advances of deep reinforcement learning (DRL) [11, 12], it has been widely investigated and employed in many fields, e.g., IoT and unmanned aerial vehicle (UAV) [13, 14], and can solve decision-making problems with high state and action dimensions, e.g., base station selection, channel selection, and caching as well as offloading decisions to maximize the long-term rewards [15]. In order to optimize the performance of MEC-based system, e.g., reducing system energy as well as computing latency and improving the social welfare as well as the quality of service (QoS) of end devices [16], DRL technique can be leveraged to allocate the resources of MEC servers, e.g., computing power and bandwidth.

Some literature has studied the implementation of mobile blockchain by leveraging the MEC and DRL techniques. Nguyen et al. [17] formulated the system privacy and system cost into two computing modes, i.e., local execution and offloading to MEC servers, and leveraged a deep Q-learning network (DQN)-based algorithm to maximize the system privacy as well as minimize the system cost. A macro base station-based framework is proposed for device-to-device content caching in [18], where DRL technique is employed to obtain the optimal content caching policy to maximize the caching resource utility. An asynchronous advantage actor-critic (A3C)-based algorithm is designed in [19] to maximize the computation rate of MEC system and blockchain throughput by optimizing the offloading decision, power allocation, block size and interval. Qiu et al. [20] jointly considered two computing modes, i.e., local execution and offloading to servers for block mining and task processing of mobile devices, and employed a deep deterministic policy gradient (DDPG)-based algorithm, named DRGO to solve the task offloading problem to minimize the system cost. However, these researches commonly considered the mining task offloading and computing power allocation, while ignoring the limited bandwidth resource of MEC servers as well as device mobility. The DRL-based solutions in [17, 18] cannot solve the formulated problem in this paper with continuous action space. The reason is that the action space of problems in these researches is discrete, and the considered DRL algorithms are value-based which cannot evaluate all the strategies (actions) and select the best one when training the neural networks in the continuous action space. Although DRGO and A3C-based algorithms both can solve decision-making problems with continuous action space, the asynchronous in A3C algorithm may reduce its performance and converge to a locally optimal solution, since the workers (copies of agents) use overdue versions of parameters, and DDPG is more suitable for the small-scale tasks compared with A3C algorithm. Thus, we choose DRGO as the benchmark method. However, DRGO still has some disadvantages to be improved, such as poor performance, convergence speed and instability. This is because the randomness of the searched actions of adaptive genetic algorithm in DRGO, and the performance of the improved actions cannot be well guaranteed. In this paper, we propose a deep reinforcement learning additional particle swarm optimization (DRPO) algorithm to solve the formulated problem by integrating DDPG and particle swarm optimization (PSO) algorithms (if the critic net of DDPG cannot evaluate an action well, i.e., the loss of the critic net is larger than a threshold, we leverage DDPG to generate actions to expand the number of training samples). Otherwise, PSO algorithm is leveraged to obtain an improved action so that DDPG can converge to a better solution, to avoid the unnecessary random search of DDPG, speed up the convergence, and improve the performance as well as stability compared with DDPG and DRGO algorithms.

In this paper, we propose an MEC-based mobile blockchain framework for security and privacy protection of mobile devices (e.g., smartphones and tablets) during transactions. Herein, each device with limited computing power acts as a miner, and the mining task of each miner is offloaded to nearby MEC servers. In order to maximize the total utility of all devices in long-term, we focus on the problem of joint computing power and bandwidth resource allocation of MEC servers and consider device mobility as well as blockchain throughput. Then, a DRPO algorithm is proposed to solve the formulated problem by decomposing it into two subproblems. The main contributions of this paper can be summarized as follows.

(1) We propose a practical MEC-based mobile blockchain framework to protect privacy and data security of mobile devices by considering the limited computing power and bandwidth of small base stations (SBSs), in which trustless mobile devices can trade with each other directly without a third party.

(2) We jointly consider the allocation problems of computing power and bandwidth of MEC servers to maximize the total utility of mobile devices in long-term, and take into account the device mobility as

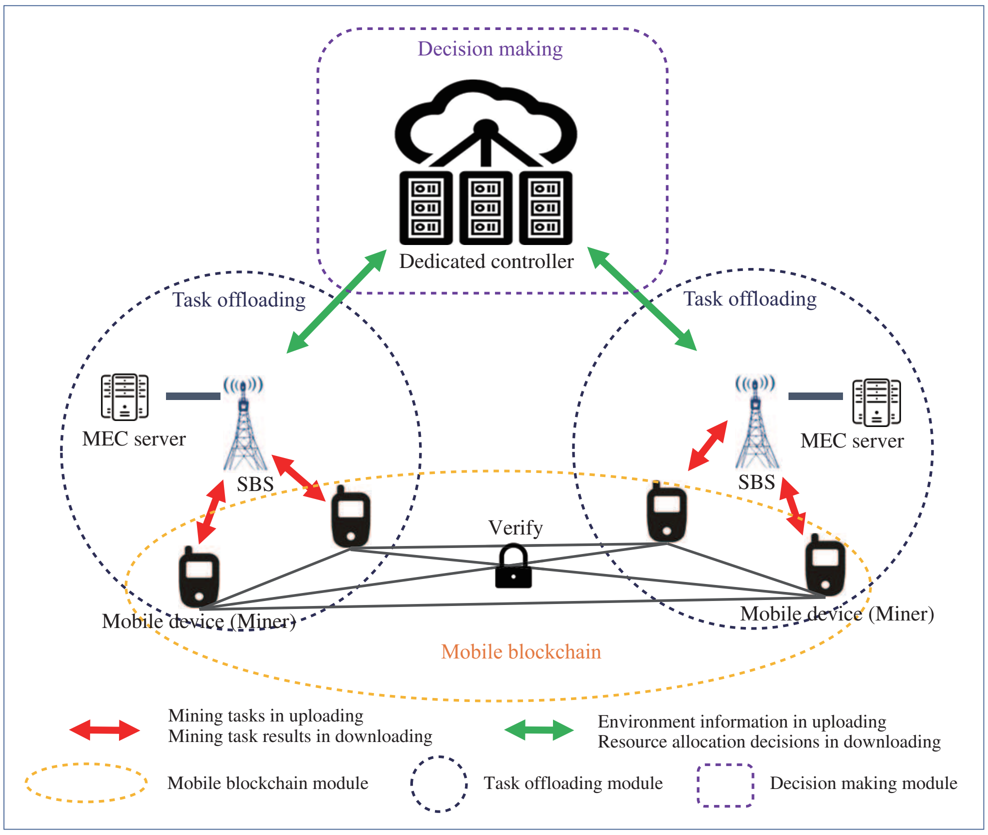

Figure 1 (Color online) System model.

well as blockchain throughput, in which the decision variables are continuous values.

(3) We propose a DRPO algorithm to obtain the optimal computing power and bandwidth allocations for each mobile device by decomposing the problem into two subproblems, which integrates the PSO scheme to avoid the unnecessary random search of DDPG, speed up the convergence, and improve the performance as well as stability.

(4) We conduct various experiments in peer edge device networks to evaluate the effectiveness of our solution. Experiment results demonstrate that our method can converge at a fast speed and maximize the total utility of all mobile devices compared with the existing ones.

The remainder of this paper is organized as follows. Section 2 elaborates the system model and the formulated problem, and a DRL-based algorithm is proposed in Section 3. The experiment results are demonstrated in Section 4. Finally, Section 5 draws the conclusion.

2 System model and problem formulation

In this section, we first elaborate the system model, followed by our formulated problem.

2.1 System model

There are three modules in our framework, i.e., mobile blockchain module, task offloading module and decision making module, as illustrated in Figure 1. Since public blockchain has the disadvantage of low throughput, and private blockchain has the flaw of centralization, this paper considers a kind of consortium blockchain, which is implemented in mobile devices to protect the privacy and data security of mobile devices during transactions. In the mobile blockchain module, all mobile devices construct a blockchain network, in which they can make transactions with each other directly, and each device acts as a miner. Owing to the limited computing capacity and energy of mobile devices, they need to offload their mining tasks to nearby MEC servers, which are deployed in SBSs for the mining rewards in the task offloading module. SBSs have limited computing power and bandwidth resources, and they need to

Table 1 Summation of main notations

Notation	Description
$\mathcal{M}$	The set of MEC servers
$\mathcal{N}_m$	The set of devices requesting from MEC server m
$\mathbb{F}_m$	The total computing power of MEC server m
$\mathbb{B}_m$	The total bandwidth of MEC server m
T_n	The mining task of device n
D_n	The original data size of mining task of device n
Y_n	The computation intensity of mining task of device n
G_n	The budget of device n for its mining task
I_n	The data size of mining result of device n
$f_{n,m}$	The allocated computing power of device n
$b_{n,m}$	The allocated bandwidth of device n
$p_{n,m}$	The unite operating price of MEC server m for device n

allocate these resources to mobile devices for solving the mining tasks. In the decision-making module, the dedicated controller can gather the whole information of SBSs as well as their corresponding mobile devices to make decisions for resource allocation (i.e., computing power and bandwidth) of each MEC server to maximize the total utility of all devices. When SBSs receive the resource allocation decisions from the dedicated controller, they need to inform each mobile device about its allocated bandwidth and computing power for payment. Then, MEC servers compute the tasks of mobile devices and return the mining results to them. Let $m \in \mathcal{M} = \{1, \ldots, M\}$ denote the set of MEC servers, and the set of mobile devices requesting services from MEC server m is $n \in \mathcal{N}_m = \{1, \ldots, N_m\}$, where M and N_m are the numbers of MEC servers and mobile devices requesting services from MEC server m, respectively. The mining task of device n is represented by $T_n = (D_n, Y_n, G_n, I_n)$, where D_n is the data size of the original mining task. We consider it equals to the data size of the mined block, since the mining task contains the whole information of the block. Y_n is the task computation intensity, i.e., the required CPU cycles, and I_n is the data size of the computation result. In this paper, we consider a more practical scenario than previous studies; i.e., each miner has its mining budget G_n, so that SBSs can allocate bandwidth and computing power based on it individually. MEC server m has limited computing and bandwidth resources. Herein, $\mathbb{F}_m$ and $\mathbb{B}_m$ are the total computing power and bandwidth of MEC server m, and $f_{n,m}$ and $b_{n,m}$ are the allocated computing power and bandwidth of device n from MEC server m, respectively. For device n whose allocated computing power is $f_{n,m}$, it needs to pay for the operating expense of MEC server m, which is divided into different levels based on the allocated computing power, i.e., $p_{n,m} = \tau(f_{\min} + \lfloor \frac{\varepsilon(f_{n,m}-f_{\min})}{f_{\max}-f_{\min}} \rfloor \frac{f_{\max}-f_{\min}}{\varepsilon})$, where $p_{n,m}$ is the unit operating price of MEC server m for device n (token per second), τ is a constant parameter, ε is the number of price levels, and $f_{\max}$ and $f_{\min}$ are the upper and lower bounds of the allocated computing power for a mobile device, respectively. For clarity, the main notations of this paper are summarized in Table 1.

Each SBS leverages the orthogonal frequency division multiplexing (OFDM) technique to transmit data when its connected devices require services. In order to offload the mining task to an MEC server, device n needs to upload its original mining task to MEC server m. We consider the state of wireless channels is time-varying which can be modeled as a Markov process, and the signal-to-noise ratios (SNRs) during the uploading of mining task and the downloading of mining result at decision epoch k can be denoted by $\mathrm{SNR}_{n,m}(k)$ and $\mathrm{SNR}_{m,n}(k)$, respectively.

Then, the task uploading rate $r^{\mathrm{up}}_{n,m}(k)$ can be represented by

$$r^{\mathrm{up}}_{n,m}(k) = b_{n,m}(k)\log_2(1 + \mathrm{SNR}_{n,m}(k)), \tag{1}$$

where $b_{n,m}(k)$ is the allocated bandwidth of device n at decision epoch k. Since each device can offload its mining task to only one MEC server, each device can leverage its whole bandwidth to download the computation result of its mining task from the MEC server. Thus, the downloading rate of mining result is

$$r^{\mathrm{down}}_{m,n}(k) = b_n\log_2(1 + \mathrm{SNR}_{m,n}(k)), \tag{2}$$

where b_n is the bandwidth of device n. Furthermore, we can obtain the task uploading time $t^{\mathrm{up}}_{n,m}(k)$ and the required time for downloading the mining result $t^{\text{down}}_{m,n}(k)$, i.e.,

$$t^{\text{up}}_{n,m}(k) = \frac{D_n}{r^{\text{up}}_{n,m}(k)}, \tag{3}$$

$$t^{\text{down}}_{m,n}(k) = \frac{I_n}{r^{\text{down}}_{m,n}(k)}. \tag{4}$$

The mining task computing time of device n in MEC server m at decision epoch k is

$$t^{\text{comp}}_{n,m}(k) = \frac{Y_n}{f_{n,m}(k)}. \tag{5}$$

Thus, the mining time of device n, i.e., the sum of task uploading time, task computing time and task result downloading time can be expressed as

$$t^{\text{mine}}_{n,m}(k) = t^{\text{up}}_{n,m}(k) + t^{\text{comp}}_{n,m}(k) + t^{\text{down}}_{m,n}(k). \tag{6}$$

The mining cost of device n consists of the task uploading cost and hiring cost of MEC servers, i.e.,

$$C_{n,m}(k) = \epsilon E_n t^{\text{up}}_{n,m}(k) + p_{n,m}(k) t^{\text{comp}}_{n,m}(k), \tag{7}$$

where E_n is the transmit power of device n, and ϵ is the cost of unit energy (token per Joule).

The rewards of a miner obtained in PoW consensus, in which each miner needs to solve a difficult puzzle, contains two parts, i.e., a fixed reward and a variable reward about the fees of all transactions related to the number of transactions in the block [21]. Obviously, the variable reward can be reflected by the data size of the block. Thus, the reward of device n obtained in the mining process of our consensus mechanism can be represented by

$$R_{n,m}(k) = \Theta_{n,m}(k)(\mathbb{R} + \eta D_n), \tag{8}$$

where $\mathbb{R}$ is the fixed reward, ηD_n is the variable reward and η is the factor of the variable reward. Variable $\Theta_{n,m}(k)$ is the probability that device n successfully mines the block at decision epoch k, which is influenced by two processes, i.e., puzzle solving process in mining and propagation process in consensus. In the puzzle solving process, the probability of a miner successfully solving the puzzle is related to its allocated computing power as well as bandwidth, i.e.,

$$\delta_{n,m}(k) = \alpha \frac{f_{n,m}(k)}{\sum_{i=1}^{M}\sum_{j=1}^{N_i} f_{j,i}(k)} + \beta \frac{b_{n,m}(k)}{\sum_{i=1}^{M}\sum_{j=1}^{N_i} b_{j,i}(k)}, \tag{9}$$

where weight parameters α and β represent the significance of the factors of the allocated computing power and bandwidth to the possibility of successfully solving a puzzle, and they meet $\alpha + \beta = 1$. In the propagation process, if the propagation time of a mined block is too long owing to its large data size, the block is very likely to be abandoned; i.e., the mined block becomes an orphan one. Considering the occurrence of successfully block mining follows a Poisson process with mean t_0 similar to [22], the probability, that a successfully mined block is orphan in the propagation process, is related to the propagation time of blocks, i.e.,

$$\xi_{n,m} = 1 - \mathrm{e}^{-\frac{1}{t_0} t^{\text{prop}}_{n,m}}, \tag{10}$$

and

$$t^{\text{prop}}_{n,m} = \phi D_n \sum_{i=1}^{M} N_i, \tag{11}$$

where ϕ is a constant parameter related to the propagation time, $\sum_{i=1}^{M} N_i$ is the total number of devices in the mobile blockchain network, and $t^{\text{prop}}_{n,m}$ is the propagation time of a block [23]. Thus, the possibility of device n successfully mining a block, i.e., appending a block to the blockchain, can be expressed as

$$\Theta_{n,m}(k) = \delta_{n,m}(k)(1 - \xi_{n,m}). \tag{12}$$

The utility of device n, i.e., the difference between its reward $R_{n,m}(k)$ and cost $C_{n,m}(k)$, can be denoted as

$$U_{n,m}(k) = R_{n,m}(k) - C_{n,m}(k). \tag{13}$$

2.2 Problem formulation

In our framework, we aim to optimize the allocation of computing power and bandwidth of all MEC servers to maximize the long-term utility of all mobile devices, i.e.,

$$\begin{aligned}
\mathrm{P}: \quad & \max_{\mathcal{B}_{N,M},\mathcal{F}_{N,M}} \lim_{K\to+\infty} \frac{1}{K}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N_m} U_{n,m}(k) \qquad (14)\\
\text{s.t.} \quad & \mathrm{C1}: \sum_{n=1}^{N_m}(f_{n,m}(k)t_{n,m}^{\mathrm{comp}}(t)) \leqslant \mathbb{F}_m,\ \forall m\in\mathcal{M},\\
& \mathrm{C2}: \sum_{n=1}^{N_m} b_{n,m}(k) \leqslant \mathbb{B}_m,\ \forall m\in\mathcal{M},\\
& \mathrm{C3}: C_{n,m}(k) \leqslant G_n,\ \forall m\in\mathcal{M},\ n\in\mathcal{N}_m,\\
& \mathrm{C4}: \frac{D_n/\ell}{\min_{\forall n}\{t_{n,m}^{\mathrm{span}}+t_{n,m}^{\mathrm{mine}}(k)+t_{n,m}^{\mathrm{prop}}\}} \geqslant \Omega,\\
& \mathrm{C5}: d_{n,m}^2+(v_n t_{n,m}^{\mathrm{mine}}(k))^2-2d_{n,m}v_n t_{n,m}^{\mathrm{mine}}(k)\cos\rho_{n,m} \leqslant \omega^2,\\
& \quad \forall m\in\mathcal{M}, n\in\mathcal{N}_m,
\end{aligned}$$

where $\mathcal{B}_{N,M}=\{b_{n,m}(k)|b_{n,m}(k)\in[b_{\min},b_{\max}]\}, k=1,\ldots,K, m=1,\ldots,M, n=1,\ldots,N_m$, and $\mathcal{F}_{N,M}=\{f_{n,m}(k)|f_{n,m}(k)\in(f_{\min},f_{\max}]\}, k=1,\ldots,K, m=1,\ldots,M, n=1,\ldots,N_m$ are decision variables, i.e., the allocated bandwidth and computing power of all devices at each decision epoch, where $b_{\min}$, $b_{\max}$, $f_{\min}$ and $f_{\max}$ are the lower and upper bounds of allocated bandwidth and computing power for a mobile device, respectively.

Constraint C1 limits the allocated computing power of all devices from one MEC server cannot exceed its total computing power at each decision epoch, and constraint C2 restricts the allocated bandwidth of devices from one MEC server cannot exceed its total bandwidth. The mining cost of each device is no more than its mining budget, as guaranteed in constraint C3. Constraint C4 ensures the blockchain throughput, i.e., the number of processed transactions per second, where ℓ is the average data size of a transaction, $t_{n,m}^{\mathrm{span}}$ is the time span from the last successfully mined block to the time when device n begins mining, and Ω is the lower bound of the blockchain throughput. In constraint C5, we consider the mobility of devices, i.e., device n moves toward a direction with speed v_n. $d_{n,m}$ is the distance between device n and MEC server m, $\rho_{n,m}$ is the angle between the direction of device moving and that of device n to MEC server m, and ω is the radius of communication range of SBSs. Constraint C5 can guarantee the mining task of each device is finished by one MEC server; i.e., there is no handover of MEC servers for solving the mining task of each device. We consider the speed of mobile devices equals to that of people's movement, and miners commonly require a short time to solve the problem puzzle. Thus, the locations of miners can be guaranteed in the communication range of SBSs during block mining; i.e., constraint C5 is practical and rational in this paper.

3 DRL-based approach

In problem P, we aim at maximizing the long-term utility of all mobile devices, whose optimization objective is equivalent to

$$\min_{\mathcal{B}_{N,M},\mathcal{F}_{N,M}} \lim_{K\to\infty}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N_m} -U_{n,m}(k), \qquad (15)$$

where

$$\begin{aligned}
-U_{n,m}(k) =& C_{n,m}(k)-R_{n,m}(k)\\
=& \frac{\epsilon E_n D_n}{b_{n,m}(k)\log_2(1+\mathrm{SNR}_{n,m}(k))}+\tau\left(f_{\min}+\left\lfloor\frac{\varepsilon(f_{n,m}(k)-f_{\min})}{f_{\max}-f_{\min}}\right\rfloor\frac{f_{\max}-f_{\min}}{\varepsilon}\right)\frac{Y_n}{f_{n,m}(k)}
\end{aligned}$$

$$-\left(\alpha\frac{f_{n,m}(k)}{\sum_{i=1}^{M}\sum_{j=1}^{N_i}f_{j,i}(k)}+\beta\frac{b_{n,m}(k)}{\sum_{i=1}^{M}\sum_{j=1}^{N_i}b_{j,i}(k)}\right)\mathrm{e}^{-\frac{1}{t_0}\phi D_n\sum_{i=1}^{M}N_i}(\mathbb{R}+\eta D_n). \tag{16}$$

We set

$$\Phi_{n,m}(k)=\tau\left(f_{\min}+\left\lfloor\frac{\varepsilon(f_{n,m}(k)-f_{\min})}{f_{\max}-f_{\min}}\right\rfloor\frac{f_{\max}-f_{\min}}{\varepsilon}\right)\frac{Y_n}{f_{n,m}(k)}, \tag{17}$$

$$\Upsilon_{n,m}(k)=-\alpha\frac{f_{n,m}(k)}{\sum_{i=1}^{M}\sum_{j=1}^{N_i}f_{j,i}(k)}\mathrm{e}^{-\frac{1}{t_0}\phi D_n\sum_{i=1}^{M}N_i}(\mathbb{R}+\eta D_n), \tag{18}$$

and

$$\Psi_{n,m}(k)=\frac{\epsilon E_n D_n}{b_{n,m}(k)\log_2(1+\mathrm{SNR}_{n,m}(k))}-\beta\frac{b_{n,m}(k)}{\sum_{i=1}^{M}\sum_{j=1}^{N_i}b_{j,i}(k)}\mathrm{e}^{-\frac{1}{t_0}\phi D_n\sum_{i=1}^{M}N_i}(\mathbb{R}+\eta D_n). \tag{19}$$

By computing the Hessian matrices of $\Phi_{n,m}(k)$, $\Upsilon_{n,m}(k)$, and $\Psi_{n,m}(k)$, we find they are not positive semidefinite matrices and these three functions are not convex. Thus, our optimization problem is a non-convex optimization problem. The optimization objective of our formulated problem is to maximize the long-term utility of all mobile devices. Once the DRL model is established in an offline way, it can output the corresponding solutions based on various input states quickly, i.e., fast response, which is suitable for the time-sensitive mobile blockchain. Therefore, we leverage a DRL-based algorithm, i.e., DDPG which has been employed in many decision-making problems with continuous action space, to allocate continuous computing power and bandwidth for mobile devices. In our framework, the resource allocation decisions are made in a dedicated controller which is integrated with all SBSs. The DDPG algorithm can be deployed in the dedicated controller to obtain the global information, and allocate the computing power and bandwidth resources of each MEC server for its requesting service devices to maximize the total utility of all devices. However, we have to deal with the challenge of leveraging DDPG, i.e., high dimension action space. The reason is twofold: (1) there are enormous number of mobile devices in the mobile blockchain system; (2) the action space is continuous, i.e., the allocated bandwidth and computing power of each mobile device are continuous values. In order to tackle the challenge, we consider the fact that the data size of a block is little in the blockchain system and commonly no more than 1 MB, which results in a low uploading time, i.e., $t_{n,m}^{\mathrm{up}}(k)$. The low uploading time has no influence on the obtained reward of device n based on (8), and in practice it is inappreciable compared with the computing time, i.e., $t_{n,m}^{\mathrm{comp}}(k)$. Thus, we approximately decompose the above problem into two subproblems, i.e. P1 and P2.

$$\mathrm{P1}:\quad \min_{\mathcal{B}_{N,M}}\ \lim_{K\to+\infty}\frac{1}{K}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N_m}\epsilon E_n t_{n,m}^{\mathrm{up}}(k)\quad \text{s.t.}\quad \text{C2 in P}. \tag{20}$$

In P1, we intend to minimize the total mining task uploading cost of all devices in long-term by allocating the bandwidth of each MEC server, which is consistent with the objective of P, i.e., maximizing the total utility of all devices. In P2, we only need to consider the allocation of the computing power of each MEC server to maximize the total long-term utility of all devices, i.e.,

$$\mathrm{P2}:\quad \max_{\mathcal{F}_{N,M}}\ \lim_{K\to+\infty}\frac{1}{K}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N_m}U_{n,m}(k)\quad \text{s.t.}\quad \text{C1, C3, C4, C5 in P}. \tag{21}$$

In order to solve P1, i.e., minimizing the long-term mining task uploading costs of all devices, we can transform the problem into minimizing the uploading costs of all devices at each decision epoch. Since each MEC server has no coupling with others when allocating their own bandwidth, P1 can be decomposed into multiple subproblems associated with each MEC server. At decision epoch k, the subproblem to be solved in MEC server m can be expressed as

$$\mathrm{P3}:\quad \min_{\{b_{n,m}(k),n\in\mathcal{N}_m\}}\Gamma_{n,m}(k)\quad \text{s.t.}\quad \sum_{n=1}^{N_m}b_{n,m}(k)\leqslant\mathbb{B}_m, \tag{22}$$

where $\Gamma_{n,m}(k)=\sum_{n=1}^{N_m}\epsilon E_n t_{n,m}^{\rm up}(k)=\sum_{n=1}^{N_m}\frac{\epsilon E_n D_n}{b_{n,m}(k)\log_2(1+{\rm SNR}_{n,m}(k))}$. It is obvious that function $\Gamma_{n,m}(k)$ is a derivative multivariate function of $\{b_{n,m}(k), n\in\mathcal{N}_m\}$, and its gradient function can be calculated by

$$\begin{aligned}\nabla\Gamma_{n,m}(k)=\Bigg(&\frac{-\epsilon E_1 D_1}{b_{1,m}^2(k)\log_2(1+{\rm SNR}_{1,m}(k))},\ldots,\frac{-\epsilon E_n D_n}{b_{n,m}^2(k)\log_2(1+{\rm SNR}_{n,m}(k))},\ldots,\\&\frac{-\epsilon E_{N_m} D_{N_m}}{b_{N_m,m}^2(k)\log_2(1+{\rm SNR}_{N_m,m}(k))}\Bigg).\end{aligned}\tag{23}$$

Furthermore, we can obtain the Hessian matrix of $\Gamma_{n,m}(k)$ since its gradient function is derivative, i.e.,

$$H_{\Gamma_{n,m}(k)}=\begin{bmatrix}\frac{2\epsilon E_1 D_1}{b_{1,m}^3(k)\log_2(1+{\rm SNR}_{1,m}(k))} & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \frac{2\epsilon E_{N_m} D_{N_m}}{b_{N_m,m}^3(k)\log_2(1+{\rm SNR}_{N_m,m}(k))}\end{bmatrix}.\tag{24}$$

Since variables ϵ, E_n, D_n, ${\rm SNR}_{n,m}(k)$ and $b_{n,m}(k)$ are positive numbers, the diagonal matrix $H_{\Gamma_{n,m}(k)}$ is a positive definite matrix. Thus, the objective function in P3, i.e., $\Gamma_{n,m}(k)$ is a strictly convex function. In addition, the constraint in P3 is an affine function. Therefore P3 is a convex optimization problem, which is not difficult to be solved.

To solve P2 which is a non-convex optimization problem, the DDPG algorithm is leveraged to allocate optimal computing power for devices to optimize the total utility. DDPG integrates deterministic policy gradient (DPG) and DQN algorithms, and can deal with the decision making problems with continuous action space. DDPG is composed of two main networks, i.e., the actor net and critic net. The actor net is responsible for the action generation, while the critic net can guide the actor net, i.e., estimating the state-action value. The critic net can evaluate the quality of the action and guide the actor net to adjust the network parameters towards a better action. Similar to the target and online nets in DQN, the actor net and critic net both have the target subnet and online subnet, and the two subnets have the same network structure.

At each decision epoch t in DDPG, we define the state as s_t, the action as a_t, and the reward function as $r(s_t,a_t)$. The action's policy of DDPG is deterministic which can be represented by $\mu:\mathcal{S}\rightarrow\mathcal{A}$, and then the recursive relationship in DRL, i.e., Bellman equation can be written as

$$Q^\mu(s_t,a_t)={\rm E}_{r_t,s_{t+1}\sim\psi}[r(s_t,a_t)+\hbar Q^\mu(s_{t+1},\mu(s_{t+1}))],\tag{25}$$

where $\hbar$ denotes the discounted factor and ψ is the expectation distribution for s_{t+1} and r_t [24].

When the function approximator is parameterized by θ^Q, the loss function of critic net used to evaluate the difference between the two sides of the Bellman equation can be expressed as

$$L(\theta^Q)={\rm E}_{s_t\sim\varrho^\varphi,a_t\sim\varphi,r_t\sim\psi}[(Q(s_t,a_t|\theta^Q)-y_t)^2],\tag{26}$$

where ϱ^φ is the distribution of state s_t under deterministic policy φ, and y_t is defined as

$$y_t=r(r_t,a_t)+\hbar Q(s_{t+1},u(s_{t+1})|\theta^Q).\tag{27}$$

The policy update of actor net needs the aid of critic net, and the policy gradient of actor net can be calculated by

$$\nabla_{\theta^\mu}J(\mu)\approx{\rm E}_{s_t\sim\varrho^\varphi}[\nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)}\nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_t}],\tag{28}$$

where $\mu(s|\theta^\mu)$ is the parameterized actor function that can determine the current action by mapping the state to a specific action and θ^μ is the variables of online actor net. Performance objective $J(\mu)$ is used to evaluate the performance of policy μ, which can be expressed as

$$J(\mu)={\rm E}_{s_t\sim\varrho^\varphi}[Q(s_t,\mu(s_t|\theta^\mu))].\tag{29}$$

In the training of DDPG, an experience replay buffer is leveraged to store the quadruple (s_t,a_t,r_t,s_{t+1}) at decision epoch t, where s_t is the current state, a_t is the current action and it commonly equals to the output action of actor net adding noise n_0 to increase the randomness of exploration, r_t is the obtained reward when executing action a_t at state s_t and s_{t+1} is the next state. During learning, mini-batch samples are randomly selected from the experience replay buffer and are input to the actor and critic nets for updating. For example, when the mini-batch contains W samples, the next state s_{q+1} obtained at decision epoch q can be input into the actor target net to achieve action to the critic target net. The critic target net can further calculate y_q in (27) to input the critic online net. The mini-batch action $a = \mu(s_q)$ can be obtained by inputting the mini-batch state s_q into the actor online net. Then the mini-batch action is input to the critic online net to generate action a's gradient, i.e., $\nabla_a Q(s, a|\theta^Q)|_{s=s_q,a=\mu(s_q)}$. Since parameter θ^μ's gradient of actor online net can be derived by its own optimizer, i.e., $\nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_q}$, the actor online net can be updated based on the following rule:

$$\nabla_{\theta^\mu} J(\mu) \approx \frac{1}{W}\sum_q [\nabla_a Q(s, a|\theta^Q)|_{s=s_q,a=\mu(s_q)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_q}]. \tag{30}$$

The update of critic online net can be finished by its own optimizer, e.g., Adamoptimizer.

Different from the update of the target net in DQN, i.e., directly copying the weights of the online net, 'soft' target update is leveraged in DDPG. Let $\mu'(s|\theta^{\mu'})$ and $Q'(s, a|\theta^{Q'})$ denote the actor and critic target nets, respectively. At first, they are a copy of the actor and critic online nets. Then, the weights of the actor and critic target nets can be updated by

$$\begin{aligned} \theta^{\mu'} &\leftarrow \varsigma\theta^\mu + (1-\varsigma)\theta^{\mu'}, \\ \theta^{Q'} &\leftarrow \varsigma\theta^Q + (1-\varsigma)\theta^{Q'}, \end{aligned} \tag{31}$$

where ς is a small constant. In this way, the target weights are constrained to change slowly, greatly improving the stability of learning [25].

In the following, we describe the three core elements of DDPG in detail, i.e., the state space, the action space and the reward function.

State space. In DDPG, the environment state S^k in decision epoch k is the union of the state of SBS S^k_m, including the offloading task of mobile device $T_n(D_n, Y_n, G_n, I_n)$, the location and moving direction as well as the speed of each device ($d_{n,m}$, $\rho_{n,m}$ and v_n, respectively), the channel state ($\mathrm{SNR}_{n,m}(k)$ and $\mathrm{SNR}_{m,n}(k)$), and the total computing power and bandwidth of SBS ($\mathbb{F}_m$ and $\mathbb{B}_m$, respectively), i.e., $S^k = \{S^k_m|m = 1, \ldots, M\}$, where S^k_m can be expressed as

$$S^k_m = \{(T_n, d_{n,m}, \rho_{n,m}, v_n, \mathrm{SNR}_{n,m}(k), \mathrm{SNR}_{m,n}(k), \mathbb{F}_m, \mathbb{B}_m)|n = 1, \ldots, N_m\}. \tag{32}$$

Action space. After each mobile device generates its task for block mining, DRL agent (i.e., the dedicated controller) needs to make allocation decisions of the computing power resource for each MEC server. Let $A^k = \{A^k_m|m = 1, \ldots, M\}$ denote the action of DRL agent, where A^k_m is the resource allocation decision for MEC server m and can be represented by

$$A^k_m = \{f_{n,m}(k)|n = 1, \ldots, N_m, f_{n,m}(k) \in (f_{\min}, f_{\max}]\}. \tag{33}$$

Reward function. In order to optimally allocate resources for mobile devices, the reward function is required to represent the objectives of our system, i.e., maximizing the total utility of all mobile devices while considering the constraints of limited resources, the mobility of devices and the blockchain throughput. We use $R^k(S^k, A^k)$ to denote the immediate reward of taking action A_k at state S^k at decision epoch k, i.e.,

$$R^k(S^k, A^k) = \begin{cases} \dfrac{1}{\lambda}\displaystyle\sum_{m=1}^{M}\sum_{n=1}^{N_m} U_{n,m}(k), & \text{if constraints C1, C3, C4, C5 in P are satisfied,} \\ 0, & \text{otherwise,} \end{cases} \tag{34}$$

where λ is a constant.

DDPG increases the randomness of exploration by adding noise to the output action of actor net, i.e., selecting a random action from a normal distribution, whose mean value is the action obtained by the actor net and variance is a given value that decreases with the number of training episodes. However, fully exploring all actions in the continuous action space is impractical, and the strict constraints of P2 make it difficult to find a satisfied feasible solution, which leads to a large number of unnecessary random search in DDPG. The mentioned reasons decrease the data learning efficiency of DDPG and consume a great amount of time to achieve convergence. In order to improve the efficiency of the exploration process to avoid unnecessary search, we introduce the PSO algorithm in the randomness exploration process of DDPG to generate an improved solution to critic net, and propose the DRPO algorithm.

In PSO, there are Z particles and each particle z has two attributes, i.e., velocity ι_z and location x_z, which represent the moving speed and direction of particle z, respectively. There are many particles in PSO, where the location of each particle represents a solution of P2 (i.e., an improved random action in DDPG) and the velocity of each particle represents the change of the solution. During iteration $\mathfrak{z}$, each particle z, whose velocity and location are $\iota_z^{\mathfrak{z}}$ and $x_z^{\mathfrak{z}}$, searches its current private optimal solution $\mathfrak{g}_z$. Then it shares its optimal solution to others so that every particle can acquire the global optimal solution $\mathfrak{G}$. Based on the private optimal solution and the global optimal solution, each particle can update its velocity $\iota_z^{\mathfrak{z}+1}$ and location $x_z^{\mathfrak{z}+1}$ following the two rules:

$$\iota_z^{\mathfrak{z}+1} = \varpi^{\mathfrak{z}}\iota_z^{\mathfrak{z}} + c_1\varkappa_1(\mathfrak{g}_z - x_z^{\mathfrak{z}}) + c_2\varkappa_2(\mathfrak{G} - x_z^{\mathfrak{z}}), \tag{35}$$

$$x_z^{\mathfrak{z}+1} = x_z^{\mathfrak{z}} + \iota_z^{\mathfrak{z}+1}. \tag{36}$$

In the former, c_1 and c_2 are learning factors, $\varkappa_1$ and $\varkappa_2$ are random numbers within $(0, 1)$, and $\varpi^{\mathfrak{z}}$ is the inertia factor satisfying $\varpi^{\mathfrak{z}} > 0$. Commonly, a larger (smaller) $\varpi^{\mathfrak{z}}$ can make the PSO algorithm have a stronger (weaker) ability to search the global optimal solution, but a weaker (stronger) ability to search the locally optimal solution. In this paper, the value of $\varpi^{\mathfrak{z}}$ is determined by leveraging the linearly decreasing weight (LDW) policy [26], i.e.,

$$\varpi^{\mathfrak{z}} = (\varpi_{\text{in}} - \varpi_{\text{out}})(\mathfrak{d} - \mathfrak{z})/\mathfrak{d} + \varpi_{\text{out}}, \tag{37}$$

where ϖ_{in} is the initial inertia factor and ϖ_{out} is the terminal inertia factor when PSO algorithm achieves the maximum number of iteration $\mathfrak{d}$. Since the velocity and location of particles are bounded, we need to redefine the update rules of velocity and location by considering their bounds, i.e.,

$$\iota^{\mathfrak{z}+1} = \begin{cases} \iota_{\min}, \text{ if the value in (35) is smaller than } \iota_{\min}, \\ \iota_{\max}, \text{ if the value in (35) is larger than } \iota_{\max}, \\ \varpi^{\mathfrak{z}}\iota_z^{\mathfrak{z}} + c_1\varkappa_1(\mathfrak{g}_z - x_z^{\mathfrak{z}}) + c_2\varkappa_2(\mathfrak{G} - x_z^{\mathfrak{z}}), \text{ otherwise,} \end{cases} \tag{38}$$

$$x^{\mathfrak{z}+1} = \begin{cases} x_{\min}, \text{ if the value in (36) is smaller than } x_{\min}, \\ x_{\max}, \text{ if the value in (36) is larger than } x_{\max}, \\ x_z^{\mathfrak{z}} + \iota_z^{\mathfrak{z}+1}, \text{ otherwise.} \end{cases} \tag{39}$$

where $\iota_{\min}$, $\iota_{\max}$, $x_{\min}$ and $x_{\max}$ are the lower and upper bounds of the velocity and the location of particles, respectively. Until the end of the iteration, an improved random action (i.e., an improved computing power allocation solution) can be generated and input into the critic net in DDPG to estimate the state-action value. It is worth noticing that the generated action of actor net is required to be input into PSO algorithm to initialize the location of one particle to guarantee the obtained action in PSO is better than the generated action in DDPG. The improved action can be stored in the experience replay buffer for future training to accelerate the convergence of DDPG. The details of the proposed DRPO algorithm can be seen in Algorithm 1, and the key idea is that if the critic net cannot evaluate an action well, i.e., the loss of the critic net is larger than a threshold, we leverage DDPG to generate actions to expand the number of training samples. Otherwise, PSO algorithm is leveraged to obtain an improved action so that DDPG can converge to a better solution.

4 Performance evaluation

Similar to [27, 28], we leverage simulations and conduct a large number of experiments to demonstrate the effectiveness of our algorithm in this section, since it is impractical to establish a large-scale mobile blockchain framework in the laboratory environment by considering the limits of the number of mobile devices and the computing capacity of desktops.

Algorithm 1 The pseudo-code of DRPO algorithm

```
Input: $\{T_n\}$, $\{\mathbb{F}_m, \mathbb{B}_m\}$;
Output: $\mathcal{B}_{N,M}, \mathcal{F}_{N,M}$;
1: Initialize the parameters in DDPG:
2:     Parameters of actor online net and critic online net, i.e., $\theta^{\mu}$ and $\theta^{Q}$;
3:     Parameters of actor target net and critic target net, i.e., $\theta^{\mu'} \leftarrow \theta^{\mu}$, $\theta^{Q'} \leftarrow \theta^{Q}$;
4:     Experience replay buffer $\mathcal{L}$ with size $l$;
5:     Number indicator $\wp$ of samples in $\mathcal{L}$;
6:     Decision epoch $k$ and constant parameter $\zeta$;
7: Initialize the parameters in PSO:
8:     Maximum number of iteration $\eth$;
9:     Number of particles $Z$;
10: while the maximum number of repetitions is not reached do
11:     PSO$_{\text{flag}}$ $\leftarrow$ False;
12:     for $k = 1; k \leqslant K; k++$ do
13:         Solve P1 to obtain the optimal bandwidth allocation strategy $\{b_{n,m}(k)\}$ at decision epoch $k$;
14:         Dedicated controller inputs system state $S^k$ to the actor net of DDPG to obtain action $A^k$;
15:         Adding noise to action $A^k$, i.e., $A^k \leftarrow A^k + n_0$;
16:         if PSO$_{\text{flag}}$ = True then
17:             Replace $A_k$ with an improved action by leveraging the PSO algorithm, i.e., $A^k \leftarrow \text{PSO}(A^k)$;
18:         end if
19:         Execute action $A^k$ (i.e., $\{f_{n,m}(k)\}$) and get reward $R^k$ as well as the next state $S^{k+1}$;
20:         Store transition quadruple $(S^k, A^k, R^k, S^{k+1})$ in experience replay buffer $\mathcal{L}$;
21:         $\wp \leftarrow \wp + 1$;
22:         if $\wp > l$ then
23:             Select $W$ mini-batch samples from $\mathcal{L}$ and update the parameters of critic online net as well as actor online net, i.e.,
                $\theta^{Q}$ and $\theta^{\mu}$ based on (26) and (30);
24:             if the loss of critic net is lower than $\zeta$ then
25:                 PSO$_{\text{flag}}$ $\leftarrow$ True;
26:             else
27:                 PSO$_{\text{flag}}$ $\leftarrow$ False;
28:             end if
29:         end if
30:         Regularly update the parameters of actor target net and critic target net according to rule (31);
31:     end for
32: end while
33: return $\mathcal{B}_{N,M}, \mathcal{F}_{N,M}$.
```

4.1 Simulation setup

We make the simulations in a 64 bit Window 10 operating system computer, which has 8 G RAM, an Intel(R) Core(TM) i7-3520M CPU with 2.90 GHz frequency, and an NVIDIA NVS 5400 M GPU. We consider a scenario that there are some SBSs in the area, and the communication radius of each SBS is 500 m (i.e., $\omega = 500$). In addition, there are 10 mobile devices requesting services from each SBS (without loss of generality, the number of mobile devices can be arbitrary), which are distributed within the overlay area of each SBS randomly (i.e., $d_n \in (0, 500)$). Without loss of generality, the average mining time for a block is 5 min (i.e., $t_0 = 300$), and in blockchain, each device can begin mining anytime (i.e., $t_{n,m}^{\text{span}} \in [0, 300)$). The transmit power of each device is 0.5 W (i.e., $E_n = 0.5$) [20]. The available bandwidth of each device is 0.5 MHz (i.e., $b_n = 0.5$), and that of each SBS is 20 MHz (i.e., $\mathbb{B}_m = 20$) [29]. The total computing power of each SBS is 10^{12} CPU cycles (i.e., $\mathbb{F}_m = 10^{12}$). The mobile devices move toward a direction with speed $[0, 1]$ m/s (i.e., $v_n \in [0, 1]$), and the corresponding angle with the SBS is between 0° and 180° (i.e., $\rho_{n,m} \in [0, 180]$).

In order to successfully mine a block, the devices need to offload their mining tasks to the corresponding SBSs. For the mining tasks, their data size is $[5, 10]$ kB (i.e., $D_n \in [5 \times 10^{-3}, 10^{-2}]$) [30]. The computing intense of each mining task is proportional to its data size, and the scale coefficient is 5×10^{10} (i.e., $Y_n = 5 \times 10^{10} \times D_n$). The data size of the mining result is 1 kB (i.e., $I_n = 10^{-3}$). We find the task uploading cost is tiny compared with the hiring cost of MEC servers and consider the data size of mining tasks D_n obeys a normal distribution [22], the mining budget of devices can be determined as $[30, 50]$ tokens (i.e., $G_n \in [30, 50]$) by estimating the upper bound of the mining cost of majority miners based on statistical analysis. We set the fixed reward for successfully mining a block as 100 tokens (i.e., $\mathbb{R} = 100$), and the coefficient of the variable reward as $10^{5.5}$ (i.e., $\eta = 10^{5.5}$). Each joule costs 10^{-4} tokens (i.e., $\epsilon = 10^{-4}$) [20], the constant parameter τ equals to 10^{-7}, the number of price level ε is 10, and the broadcast factor is 0.5 (i.e., $\phi = 0.5$) [23]. The average data size of each transaction is 0.2 byte (i.e., $\ell = 2 \times 10^{-7}$), and the lower bound of blockchain throughput is 100 (i.e., $\Omega = 100$). The minimum and

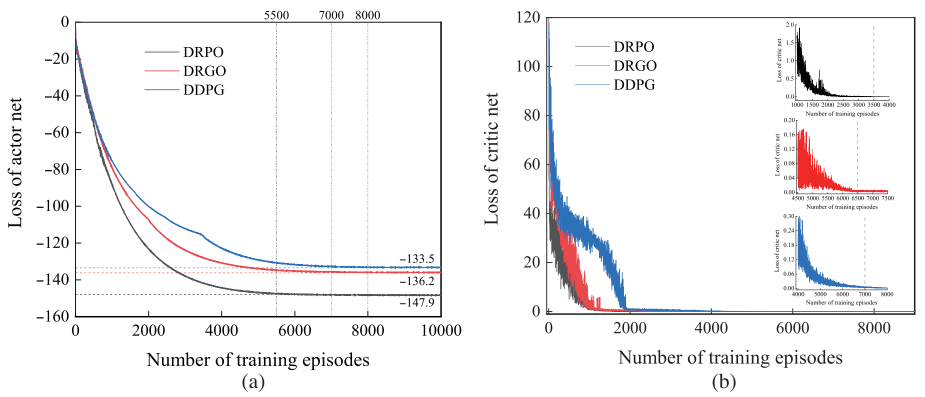

Figure 2 (Color online) The convergence performance of different methods: (a) the loss of actor net, and (b) the loss of critic net.

maximum values of the allocated bandwidth and computing power are 0.02, 2, $\frac{1}{3}\times 10^6$ and $\frac{2}{3}\times 10^{12}$ (i.e., $b_{\rm min}=0.02, b_{\rm max}=2, f_{\rm min}=\frac{1}{3}\times 10^6, f_{\rm max}=\frac{2}{3}\times 10^{12}$), respectively.

The parameters in DRPO are designed as follows. There is one hidden layer in the actor net, and the number of the neurons of the hidden layer is related to the number of mobile devices as well as the dimension of the input data (e.g., if the number of devices is 10, the number of neurons of the hidden layer is 260). The activation function of the hidden layer is error linear unit (ELU), and that of the output layer is the sigmoid function. The learning rate of the actor net is 0.001, and that of the critic net is 0.002. The constant in 'soft' target update is 0.01 (i.e., $\varsigma=0.01$). The size of the experience replay buffer is 1000 (i.e., $l=1000$), and the batch size in training is 32 (i.e., $W=32$). The constant in reward function is 200 (i.e., $\lambda=200$).

For the parameters of PSO in DRPO, the number of particles is 10 (i.e., $Z=10$), the initial inertia factor is 0.9 and the terminal inertia factor is 0.4 (i.e., $\varpi_{\rm in}=0.9$, $\varpi_{\rm out}=0.4$). We set the lower and upper bounds of the velocity and location to $-3\times 10^{10}, 3\times 10^{10}, \frac{1}{3}\times 10^6$ and $\frac{2}{3}\times 10^{12}$ (i.e., $\iota_{\rm min}=-3\times 10^{10}, \iota_{\rm max}=3\times 10^{10}, x_{\rm min}=\frac{1}{3}\times 10^6, x_{\rm max}=\frac{2}{3}\times 10^{12}$), respectively. The number of particles in the benchmark method PSO is 200, and the maximum number of iterations is 500.

4.2 Simulation design

Since the objective of this paper is to maximize the total utility of all mobile devices in long-term, the evaluation metric of our simulation is the average total utility of all devices in each decision epoch, i.e., $\frac{1}{K}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N_m}U_{n,m}(k)$. In the simulation, K is set to 5 by considering the fact that a large value of K leads to a long training time in the DDPG-based algorithm.

Three representative baseline algorithms are leveraged to compare with our DRPO algorithm.

- PSO algorithm. It is a heuristic algorithm, which is commonly used to search solutions with high quality for the problems with large search space.
- DDPG. It is the state-of-art DRL scheme to search for the optimal solution for the problem with continuous action space [25].
- DRGO. It is an improved DDPG algorithm, which integrates DDPG with an adaptive genetic algorithm to speed up the convergence of DDPG caused by the high-dimension action space [20].

4.3 Performance analysis

In this paper, for each group of the experiment, we repeat it 20 times and take the average value of all results as the final experiment result.

4.3.1 *Convergence performance*

We first evaluate the convergence property of different algorithms since it is one of the most significant factors for the effectiveness of DRL. Under the scenario that the parameters are set as described in Subsection 4.1 and the number of mobile devices is 10, the losses of actor net and critic net at each training episode for various methods are illustrated in Figures 2(a) and (b), respectively. From Figure 2(a), it is obvious that the loss of actor net in DRPO is lower than that of DRGO and DDPG, and the required

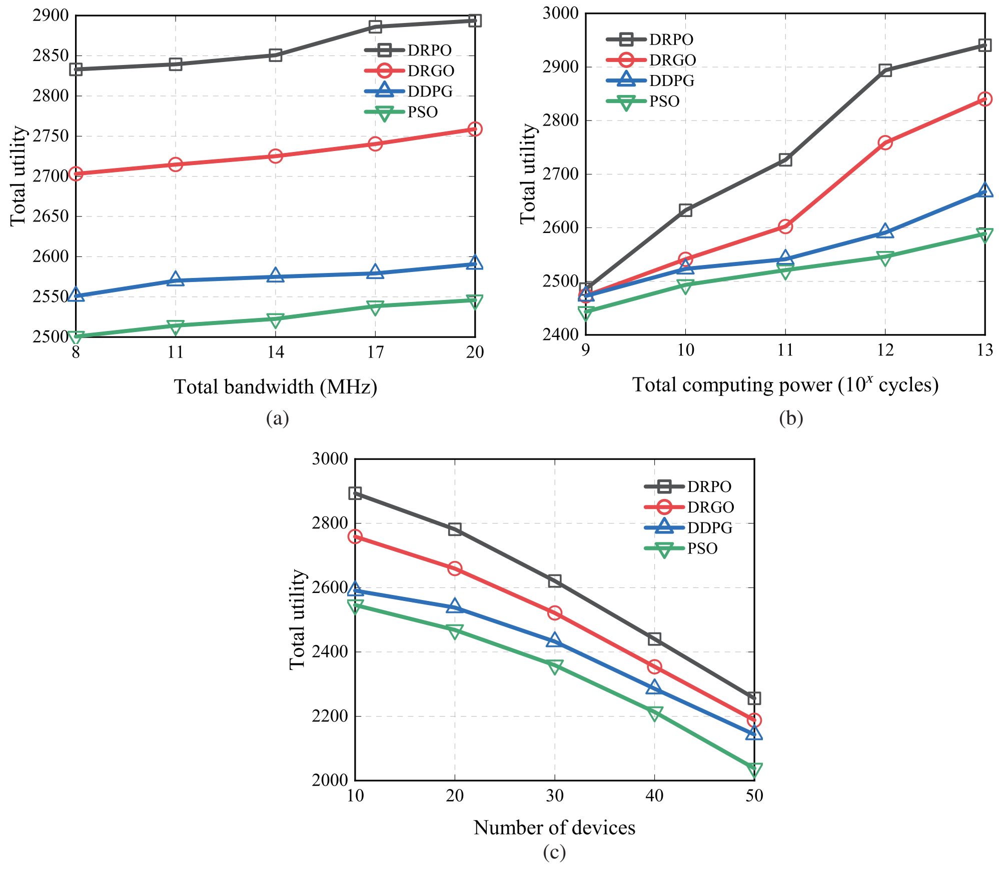

Figure 3 (Color online) Experiment results for different: (a) bandwidths, (b) computing powers, and (c) numbers of devices.

number of training episodes for convergence in DRPO, DRGO and DDPG is 5500, 7000 and 8000, respectively. It demonstrates that DRPO can generate a better action than DRGO and DDPG, and converge to the optimal action faster. In Figure 2(b), we can find that the loss fluctuation of critic net in DRPO is smaller than that of DRGO and DDPG, and the critic net of DRPO can reach convergence faster than DRGO and DDPG, which illustrates that DRPO can evaluate actions better (i.e., more stable) and converge to the maximal reward faster than DRGO and DDPG.

4.3.2 *Performance under different bandwidths*

When the number of devices is 10 and the total bandwidth of each SBS ranges from 8 to 20, the total utility under different bandwidths is illustrated in Figure 3(a). It is obvious that the total utility of all mobile devices increases with the increase of bandwidth, since more bandwidth can be allocated to each mobile device, which leads to lower task uploading time and less task uploading cost. The performance of the DRL-based algorithm is better than that of PSO, since PSO is a heuristic algorithm and easy to converge to a locally optimal solution. However, the growth extent of total utility is small, because bandwidth allocation has little influence on our optimization objective as explained in Section 3. It also demonstrates the effectiveness of our problem decomposition. For DRPO, it can generate higher utility than DDPG, and the reason is that, when the actor net can evaluate actions well, DRPO can improve the generated solutions of DDPG by leveraging PSO algorithm. The performance of DRPO is better than that of DRGO. This is because PSO can effectively search for a better solution faster in the continuous action space compared with the adaptive genetic algorithm, by memorizing and sharing the historical optimal solutions among all particles to search an optimal action in each iteration, other than the blind search (e.g., the crossover and mutation in the genetic algorithm) in the whole solution space.

4.3.3 *Performance under different computing powers*

For different computing powers of SBSs, the total utility of all devices is illustrated in Figure 3(b). It can

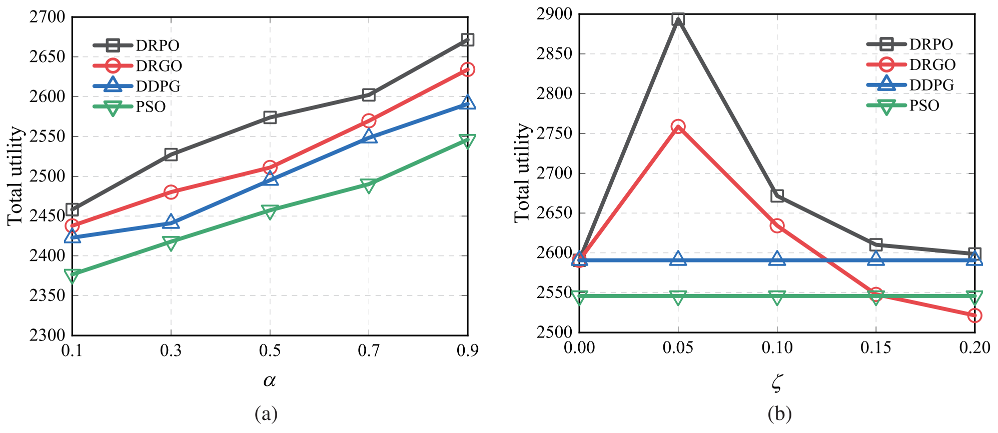

Figure 4 (Color online) Experiment results under: (a) different values of α, and (b) different values of ζ.

be seen that when the total computing power of SBSs ranges from 10^9 to 10^{13} cycles, the total utility increases. The reason is that the increasing computing power of SBSs provides a strong computing capacity for mobile devices, which can reduce the computing time and hiring cost enormously, although the unit operating price of MEC servers may increase. Based on the obtained total utility, the performance comparison result of the four algorithms is DRPO>DRGO>DDPG>PSO. The performance of DDPG is better than PSO, since the DRL-based algorithm has a good exploration ability and can find a satisfied optimal solution by training, but the performance of PSO relies on initial solutions, and it is easy to converge to a locally optimal solution. DRGO obtains a higher total utility compared with DDPG, because DRGO can improve the generated solutions of DDPG by leveraging the adaptive genetic algorithm. The reason that the performance of DRPO is better than DRGO has been explained before.

4.3.4 *Performance under different numbers of devices*

For the different number of mobile devices, the total utility of all devices is illustrated in Figure 3(c), from which we can see that the obtained utility of all mobile devices decreases as the increasing number of mobile devices. This is because the increasing number of devices decreases the allocated bandwidth and computing power of each mobile device, which can reduce the utility of each mobile device enormously, even the number of devices increases. In addition, the obtained total utility of DRPO is larger than that of the benchmark methods while the performance of PSO is worst when the number of devices ranges from 10 to 50. The performance gaps among DRPO, DRGO and DDPG decrease as the increasing number of mobile devices. The reason is that the increasing number of devices enlarges the dimensions of state and action spaces in these three DRL-based methods, and increases the difficulty of solving the formulated problem. Although the three schemes can find satisfied solutions, their obtained total utility decreases sharply when the number of mobile devices increases.

4.3.5 *Performance under different values of α*

Parameter α is used to evaluate the significance degree of the allocated computing power of devices to the probability of successfully solving a puzzle, and it also influences the total utility of all mobile devices. Thus, we intend to select a satisfying value for α based on the obtained total utility. For different values of α, the obtained total utility of the four methods is illustrated in Figure 4(a). From this figure, we can observe that the performance of DRPO is always better than that of the benchmark methods, which demonstrates its effectiveness. In addition, the performance of the four schemes increases as the increase of α. In order to obtain a larger value for our optimization objective, we need to pay more attention to the allocated computing power, which also demonstrates the effectiveness of our problem decomposition. Thus, in the experiment, we set the value of α to 0.9.

4.3.6 *Performance under different values of ζ*

In DRPO, variable ζ has a direct relationship with the trade-off between the improved actions of PSO and the generated actions of DDPG. From Figure 4(b), it is obvious that when the value of ζ is between 0 and 0.05, the performance of DRPO and DRGO increases with the increase of ζ, since DRPO and

DRGO have a larger probability to improve actions by leveraging PSO and adaptive genetic algorithms, respectively. When the value of ζ is a larger than 0.05, the total utility of all devices in DRPO decreases as the increase of ζ. The reason is that a larger ζ can make DRPO algorithm have a higher probability to improve actions by leveraging PSO algorithm, and vice versa. Since the performance of PSO relies on the initialization of solutions, and the performance difference of the improved action each time may be large, which can influence the learning effect, learning stability and convergence speed of DRPO, and finally lead to a worse solution. The performance trend of DRGO is similar to DRPO. When $\zeta = 0$, the performance of DRPO and DRGO is the same with DDPG, because there is no improvement for actions in DRPO and DRGO, and they are equivalent to DDPG algorithm. Since ζ has no influence on the performance of DDPG and PSO, their obtained utility is a fixed value. Thus, in the experiment, we set the value of ζ to 0.05.

5 Conclusion

In this paper, we design an MEC-based mobile blockchain framework to protect the privacy and data security of mobile devices when they make transactions in the IIoT system. We formulate a joint bandwidth and computing resource allocation problem to maximize the total utility of all mobile devices in long-term, and consider the mobility of devices as well as the blockchain throughput. In order to solve the formulated problem, we propose a DRPO algorithm that integrates DDPG and PSO algorithms. In DRPO, we decompose the formulated problem into two subproblems to reduce the dimension of the action space, and PSO scheme is integrated to avoid the unnecessary random search for actions in DDPG. To illustrate the effectiveness of our algorithm, we conduct enormous experiments, and the corresponding results show that DRPO can converge at a faster speed, obtain larger utility, and has stronger stability compared with the state-of-art methods of PSO, DDGP and DRGO. In future work, we will consider moving vehicles into our mobile blockchains, and solve the problems of SBS handover and task migration.

Acknowledgements This work was supported in part by National Key R&D Program of China (Grant No. 2018YFE0206800), National Natural Science Foundation of China (Grant Nos. 61701406, 61971084, 62001073), National Natural Science Foundation of Chongqing (Grant Nos. cstc2019jcyjcxttX0002, cstc2019jcyj-msxmX0208), and Chongqing Talent Program (Grant No. CQYC2020058659).

References

1 Ning Z, Dong P, Wang X, et al. Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks. IEEE Trans Mobile Comput, 2020. doi: 10.1109/TMC.2020.3025116
2 Xu L D, He W, Li S. Internet of Things in industries: a survey. IEEE Trans Ind Inf, 2014, 10: 2233–2243
3 Ning Z, Dong P, Wang X, et al. Mobile edge computing enabled 5G health monitoring for Internet of Medical Things: a decentralized game theoretic approach. IEEE J Sel Areas Commun, 2021, 39: 463–478
4 Liu D, Alahmadi A, Ni J, et al. Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain. IEEE Trans Ind Inf, 2019, 15: 3527–3537
5 Wang X, Ning Z, Zhou M C, et al. Privacy-preserving content dissemination for vehicular social networks: challenges and solutions. IEEE Commun Surv Tut, 2019, 21: 1314–1345
6 Yu Y, Ning Z L, Guo L. A secure routing scheme based on social network analysis in wireless mesh networks. Sci China Inf Sci, 2016, 59: 122310
7 Yang Z, Yang K, Lei L, et al. Blockchain-based decentralized trust management in vehicular networks. IEEE Internet Things J, 2019, 6: 1495–1505
8 Ali M S, Vecchio M, Pincheira M, et al. Applications of blockchains in the Internet of Things: a comprehensive survey. IEEE Commun Surv Tut, 2019, 21: 1676–1717
9 Ning Z, Zhang K, Wang X, et al. Intelligent edge computing in Internet of vehicles: a joint computation offloading and caching solution. IEEE Trans Intell Transp Syst, 2020. doi: 10.1109/TITS.2020.2997832
10 Wang X, Ning Z, Guo S, et al. Imitation learning enabled task scheduling for online vehicular edge computing. IEEE Trans Mobile Comput, 2020. doi: 10.1109/TMC.2020.3012509
11 Ning Z L, Zhang K Y, Wang X J, et al. Joint computing and caching in 5G-envisioned internet of vehicles: a deep reinforcement learning-based traffic control system. IEEE Trans Intell Transp Syst, 2020. doi: 10.1109/TITS.2020.2970276
12 Wang X, Ning Z, Guo S. Multi-agent imitation learning for pervasive edge computing: a decentralized computation offloading algorithm. IEEE Trans Parall Distrib Syst, 2020, 32: 411–425
13 Zhu J, Song Y, Jiang D, et al. A new deep-Q-learning-based transmission scheduling mechanism for the cognitive Internet of Things. IEEE Internet Things J, 2018, 5: 2375–2385
14 Luong N C, Hoang D T, Gong S, et al. Applications of deep reinforcement learning in communications and networking: a survey. IEEE Commun Surv Tut, 2019, 21: 3133–3174
15 Lei L, Xu H, Xiong X, et al. Multiuser resource control with deep reinforcement learning in IoT edge computing. IEEE Internet Things J, 2019, 6: 10119–10133
16 Chen M, Hao Y. Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J Sel Areas Commun, 2018, 36: 587–597
17 Nguyen D, Pathirana P, Ding M, et al. Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning. 2019. ArXiv:1908.07467

18 Dai Y, Xu D, Maharjan S, et al. Blockchain and deep reinforcement learning empowered intelligent 5G beyond. IEEE Network, 2019, 33: 10–17

19 Feng J, Yu F R, Pei Q, et al. Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: a deep reinforcement learning approach. IEEE Internet Things J, 2020, 7: 6214–6228

20 Qiu X, Liu L, Chen W, et al. Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing. IEEE Trans Veh Technol, 2019, 68: 8050–8062

21 Xiong Z, Feng S, Niyato D, et al. Edge computing resource management and pricing for mobile blockchain. 2017. ArXiv:1710.01567

22 Xiong Z, Feng S, Wang W, et al. Cloud/fog computing resource management and pricing for blockchain networks. IEEE Internet Things J, 2019, 6: 4585–4600

23 Kang J, Xiong Z, Niyato D, et al. Toward secure blockchain-enabled internet of vehicles: optimizing consensus management using reputation and contract theory. IEEE Trans Veh Technol, 2019, 68: 2906–2920

24 Qiu C, Hu Y, Chen Y, et al. Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications. IEEE Internet Things J, 2019, 6: 8577–8588

25 Lillicrap T, Hunt J, Pritzel A, et al. Continuous control with deep reinforcement learning. 2015. ArXiv:1509.02971

26 Mao C, Lin R, Xu C, et al. Towards a trust prediction framework for cloud services based on PSO-driven neural network. IEEE Access, 2017, 5: 2187–2199

27 Asheralieva A, Niyato D. Learning-based mobile edge computing resource management to support public blockchain networks. IEEE Trans Mobile Comput, 2020. doi: 10.1109/TMC.2019.2959772

28 Yu J, Kozhaya D, Decouchant J, et al. RepuCoin: your reputation is your power. IEEE Trans Comput, 2019, 68: 1225–1237

29 Liu Y, Yu F R, Li X, et al. Decentralized resource allocation for video transcoding and delivery in blockchain-based system with mobile edge computing. IEEE Trans Veh Technol, 2019, 68: 11169–11185

30 Liu M, Yu F R, Teng Y, et al. Computation offloading and content caching in wireless blockchain networks with mobile edge computing. IEEE Trans Veh Technol, 2018, 67: 11008–11021