Wireless Networks
https://doi.org/10.1007/s11276-020-02444-7

Task offloading based on deep learning for blockchain in mobile edge computing

Chung-Hua Chu[1]

Abstract
Blockchain is an advanced technique to realize smart contracts, various transactions, and P2P crypto-currencies in the e-commerce society. However, the traditional blockchain does not consider a mobile environment to design a data offloading of the blockchain such that the blockchain results in high computational cost and huge data propagation delay. In this paper, to remedy the above problem, we propose a scalable blockchain and a task offloading technique based on the neural network of the mobile edge computing scenario. Experimental results show that our approach is very scalable in the mobile scenario.

Keywords Task offloading blockchain mobile · Edge computing · Neural network · Light-weight blockchain

1 Introduction

With rapid advances in blockchain technologies, users can conveniently process digital transactions and smart contracts. Bitcoin [1] based on the blockchain presented a P2P (Peer to Peer) electronic payment system to store all digital transactions in linked blocks as public ledgers in each node. There are several existing works on the scalable and light-weight blockchain for IoT (Internet of Things). Hashemi et al. [2] and Christidis et al. [3] proposed blockchain-based algorithms for IoT while they still did not solve the computational difficulty of POW (proof-of-work) in IoT devices. Although Dorri et al. [4] presented a light-weight blockchain for IoT, the performance was still poor due to the manner of its complex related transactions. On the other hand, Xiong et al. [5] involved an edge computing resource management and pricing to support POW in the mobile blockchain. However, Xiong's method leads to huge extra cost for each transaction fee and thereby loses the advantage of the digital transaction. In recently, Bruce [6] proposed mini-blockchain while it had to maintain a complex tree to deal with balance for each account. Additionally, trust-chain [7] can offer scalability for mobile devices while it adopts a simple consensus algorithm instead of POW to verify the blocks.

The large size of the blockchain is infeasible for downloading or propagating under the large scale of mobile users. In addition, the large size of the blockchain wastes wireless network bandwidth. These issues are composed of minimizing the size of the blockchain and supporting concurrent accesses by mobile users. In this paper, we redesign the blockchain to fit the scalability and safety for the mobile network. We propose the micro-blockchain which adopts a light-weight data structure of the block for the mobile scenario. Specifically, in the micro-blockchain, a miner can get reward after solving successful puzzles. We adopt a random number and hash functions to secure the micro-blockchain. Instead of tracing the related transactions on the blockchain, the micro-blockchain is based on an account tree [6] to maintain the balance of each account. The account tree stores a unique ID on the SIM card of the mobile user. Therefore, the proposed micro-blockchain is more feasible for the mobile blockchain.

To solve the computational limits on the mobile devices, we introduce Mobile Edge Computing for solving the POW of the micro-blockchain. In this paper, we consider the Block Data Offloading Problem. We propose a novel block

✉ Chung-Hua Chu
chchu@gm.nutc.edu.tw

[1] Department of Multimedia Design, National Taichung University of Science and Technology, Taichung, Taiwan, ROC

Published online: 13 August 2020

data offloading technique referred to as a BDO (Block Data Offloading). BDO is based on the deep learning [8] to determine the optimal offloading strategy for the mobile edge computing. The deep learning based on neural networks is applied to feature recognition. It adopts stochastic gradient descent to search globally optimal solutions. Specifically, we construct a multi-layer neural network architecture to model the optimal resource management of the mobile edge computing. We use utility between the mobile users and the mobile edge units as the data training. The training data can adjust weights of the neural networks to optimize the computing and the energy resource management in the mobile environment.

We summarize the contributions of this paper as follows.

- To release constraints of the traditional blockchain, we redesign the micro-blockchain for the mobile scenario. Different from the traditional blockchain, our approach reformulates the account tree to reduce unnecessary branches on the tree. Therefore, the pruned tree can accelerate the maintainability and the traceability on the tree. The pruned tree is called the unique account tree.
- Since the unique account tree stores the balance of each account, it can reduce the complex relationship of each transaction. Additionally, to reduce large storage, the unique account tree prunes old blocks.
- To solve the complicated puzzle of POW on the mobile scenario, we devise an effective task offloading algorithm by using Mobile Edge Computing. We adopt a deep learning technique to optimize the utility between the mobile miners and the edge computations. Therefore, the mobile users can easily verify the block under the limit of computing resource on the mobile network. The light-weight micro-blockchain is robust against data loss and scarce bandwidth in the mobile network.
- We propose a framework based on the edge computing to minimize the propagation delay and the response time of each transaction in the mobile scenario.
- To effectively mitigate the transaction load, we propose a dynamic throughput adjustment to dynamically adjust throughput.

The rest of this paper is organized as follows. Preliminaries are given in Sect. 2. In Sect. 3, we develop proposed methods. Experimental results are presented in Sect. 4. Finally, this paper concludes with Sect. 5.

2 Preliminaries

2.1 Related works

In 2008, Nakamoto first presented the blockchain-based Bitcoin [1], which was a decentralized cryptocurrency on a P2P network. There are transactions representing monetary transitions with ownership information including the currency transfer among nodes on the network. In fact, the transactions in cryptocurrencies are modeled as the data structures that include the monetary value transferred among accounts. There are many cryptographic techniques to protect the transactions. Users' transactions and ledgers are stored on the untrustful nodes without a central trusted server while this cryptocurrency is trusted by every node within the system. With the hash of the prior block in the blockchain, each block packets many transactions and is linked back to its previous block. The blockchain is based on Merkle trees [9] to efficiently hash multiple transactions in the block. The Merkle trees protect the transactions from change because tampering one transaction causes the change of the root Merkle hash and the invalidation of the block. The blockchain provides an immutable and secure transaction data storage. To prevent tampering and double spending, the transaction data are not allowed to be updated or deleted on the blockchain. To verify a valid block, the node has to solve a computational difficult problem, i.e., the proof-of-work puzzle or Proof of Stake (POS) [10]. Such a computational process is called mining, and the node to run the process is named as a miner. The security of blockchain directly relies on the distributed consensus mechanism maintained by these miners [11].

There were some works on the mobile blockchain. Xiong et al. proposed a two-stage Stackelberg game model to find the best pricing strategy for offloading the POW tasks between the edge computing provider and mobile miners [5]. Stanciu et al. devised a hierarchical distributed control system using Hyperledger Fabric blockchain [12]. Samaniego et al. [13] concluded that the fog of blockchain was better than the cloud as considering network latency for IoT networks.

The latest version of blockchain allowed user to program various application with a smart contract. Omohundro et al. [14] introduced the smart contract as autonomous programs running over the blockchain. The smart contract enables programmable transactions using event triggers, conditions, and business logic. Therefore, the blockchain with the smart contract is an emerging currency transaction.

2.2 System architecture

Architecture of a scalable blockchain based on edge computing in the mobile environment is presented in Fig. 1. Mobile users get together with their friendship in the social community. Each social community consists of a large number of the mobile users. The mobile users use blockchain APPs to process currency transactions and to store the blocks in their social community. We use an edge computing service to offload the computational tasks of the blockchain applications such as POW and transaction security for each mobile device. The computational tasks of the mobile users can be offloaded to a nearby edge computing unit in their social community. Each link between the mobile devices and edge computing units is protected by public key infrastructure (PKI). Therefore, we can assume that the blockchain is maintained and is verified by each mobile device in each social community. In addition, each edge computing unit also manages transactions generated by the mobile users in each social community.

3 Proposed methods

We develop a neural network model for the offloading schedule. The neural network model is implemented to maximize the utility of the mobile users. The training procedure can be precisely close to the optimal utility of the mobile users.

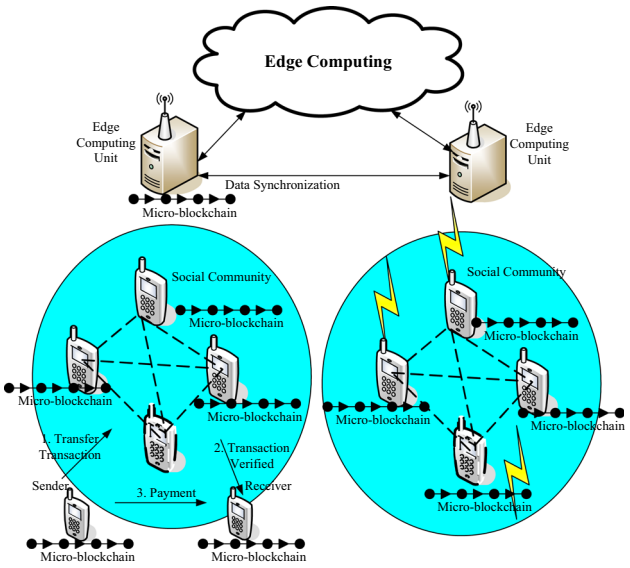

Fig. 1 Architecture of scalable blockchain based on edge computing in the mobile social network

3.1 Micro-blockchain

3.1.1 Data structure

In this Section, we discuss the data structure of the proposed micro-blockchain for mobile devices in the mobile social network. The data structure of transaction data is shown in Fig. 2. In the cryptocurrency system, the most important thing is to maintain balances of all mobile users. The traditional blockchain needs to trace all related transaction data stored in each block for obtaining balances of all mobile users. Such a data structure is computationally overhead along with the large size of the blockchain for mobile devices. Additionally, the traditional transaction data are linked each other such that they increase the computational cost to calculate balances of all mobile users. Therefore, we modify an account tree [6] to record the balances for mobile users. Note that the account tree uses a binary radix tree structure to record balances of the mobile users. The account tree can simplify the complicated transaction process by just updating balances of the mobile users in terms of the recent transactions. Therefore, the old transaction data do not need to be stored on the micro-blockchain.

The traditional account tree records balances of the addresses for the ownership of the cryptocurrency. When the size of the account tree increases, the costs to store and to trace the balances of accounts also dramatically increase. Specifically, the storage of each account needs 40 bytes for both public key and balance. If there are 5 million mobile users, each user would need to store 200 MB of ledger data as shown in Fig. 3(a). If each mobile user has 5 more accounts as shown in a gray subtree of Fig. 3(a), the storage size of 1GB would not be afforded for generic mobile devices. The proposed modified account tree prunes the branches of duplicate accounts as shown in Fig. 3(b). Then, the proposed modified account tree links the leaf nodes of the pruned branches to the node of the SIM-ID account as shown in Fig. 3(b). Therefore, the pruned tree can accelerate the maintainability and the traceability on the tree. Experimental results show that the proposed modified account tree is faster than the traditional account tree in the balance search of accounts. This is because lots of pruned branches do not need to be traced to the balance of each account in the proposed modified account tree.

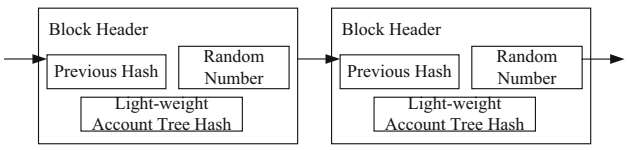

Fig. 2 Data structures of block headers in the micro-blockchain

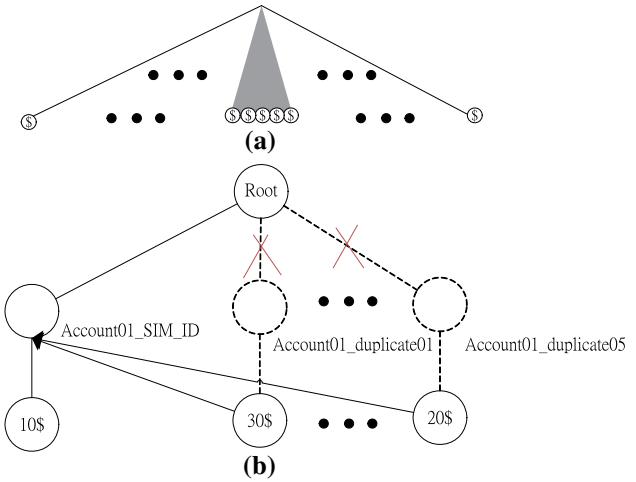

Fig. 3 Account trees (**a**) a traditional account tree (**b**) a subtree of the proposed modified tree with pruning the branches of duplicate accounts from a gray subtree of Account01 in (**a**)

Instead of storing all blocks in the traditional blockchain, we propose the data structure of the micro-blockchain to merely store the chain of block headers. This is because the micro-blockchain adopts the light-weight account tree to record the balance of each mobile user instead of linking to transaction data to compute the balance. Therefore, the micro-blockchain can reduce the large size of the conventional blockchain. Each block header embeds the root hash of the light-weight account tree and the hash of previous block header. Therefore, we can ensure that the balance of each user cannot be altered in each block header since the current block header records the hash of previous block header. We thereby can verify each block header in the micro-blockchain from the start to the end on the chain of the block header. If an attacker attempts to tamper a previously stored balance tree, the hash of the corresponding block header stored in the next block header will not be consistent. Therefore, the micro-blockchain can guarantee a cumulative difficulty of the chain to protect the balance of each user from generating a fake one.

3.1.2 Transaction verification and consensus

When the edge computing unit receives a transaction, it identifies whether the sender of this transaction belongs to its own social community. Next, the edge computing unit finds out IDs of sender and receiver from the transaction data. Then, the edge computing unit will process the transaction if the receiver belongs to its own social community. Otherwise, the edge computing unit will broadcast the transaction to all other edge computing units. Each edge computing unit stores all pending transactions while the maximum number of the pending transactions is set as N. Finally, we use POW to generate a new block header.

Each edge computing unit stores the transaction successful ratio for each mobile user in each social community. Note that the transaction successful ratio of a mobile user is a ratio that is equal to the number of valid transactions over the total number of transactions issued by the corresponding user. Next, a miner randomly chooses an edge computing unit to generate a new block header. Then, the chosen edge computing unit verifies and generates the new block header. To update the light-weight account tree in the block header, the edge computing unit finally validates the new block header and verifies each individual transaction. To valid the block header, all transactions involving the block header must be valid. Note that the edge computing unit checks the signatures of the sender and verifies each transaction. We explain each step during the transaction processing and the consensus processing of the micro-blockchain as follows.

Step 1 The mobile user, named Sender, sends a registration request with his/her SIM card ID and balances to an edge computing unit. Note that all messages are protected by using the public-key cryptography.

Step 2 The edge computing unit verifies the SIM card ID of Sender after receiving the registration request from Sender. If the identification of Sender is validated, the edge computing unit records the SIM card ID and the balance of Sender in a light-weight account tree. Otherwise, the edge computing unit will reject the registration. Then, the edge computing unit sends the result of the identification verification to Sender.

Step 3 Sender can send a transaction request to the edge computing unit. Note that the transaction request includes the SIM card ID of Sender and transaction data.

Step 4 When the edge computing unit receives a transaction, it justifies whether Sender of this transaction is validly identified. The edge computing unit will send block verification to a miner until the maximum number of the pending transactions reaches N. Note that the edge computing unit chooses the miner in terms of the largest transaction successful ratio among miners.

Step 5 For generating a new block header, the miner sends an offloading request to the edge computing unit. The edge computing unit validates the new block header and verifies each individual transaction. Then, the edge computing unit updates the light-weight account tree in the block header. After generating the new block header, the edge computing unit appends this block header in the micro-blockchain.

Steps 6–7 To update the micro-blockchain in the ledgers, the edge computing unit broadcasts the new block header to all mobile users and other edge computing units.

Figure 4 shows the flowchart of above steps.

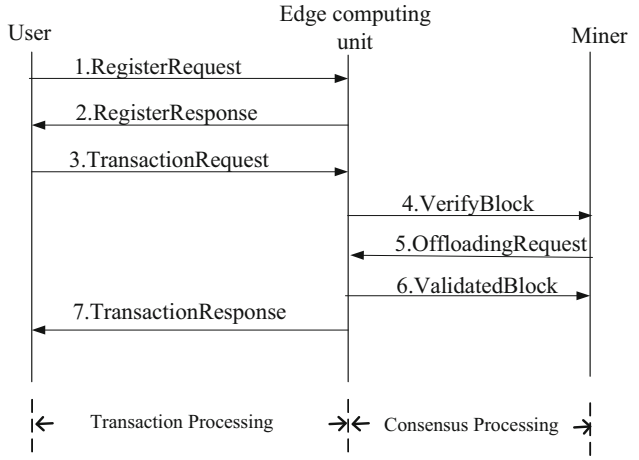

Fig. 4 Interactive message flowchart in the transaction processing and the consensus processing

3.2 Neural network mechanism

3.2.1 Notation and problem formulation

We consider that there are I mobile users in each area covered with one edge computing unit. The i-th mobile user has her/his own computing ability denoted by c_i. Note that the computing ability is relied on the CPU frequency in the mobile devices. The power consumption of the i-th mobile user for mining a block is defined as p_i. The variable reward obtained by the i-th mobile user for mining a block is symbolized by r_i. The time delay for locally mining a block on the ith mobile user is represented by t_i. Without loss of generality, we assume that each block has the same size and computing cost. The edge computing unit determines whether the block of the i-th mobile user should be processed with the edge computing. Generally speaking, when t_i, p_i, r_i of the i-th mobile user are increasing, the probability of processing the block on the edge computing unit is also increasing. This means that the mobile users have less local computing resource to generate the block. Therefore, the edge computing unit needs to help the user to process each block within consensus time. On the contrary, when c_i of the i-th mobile user is increasing, the user has his/her own computing resource to generate the block. Let u_i denote the offloading demand of the i-th mobile user for the edge computing resource expressed as

$$u_i = \frac{t_i \times p_i \times r_i}{c_i}. \tag{1}$$

Note that Eq. (1) of this paper is derived from Eqs. (3)–(6) [15], which are sufficiently proved by mathematical analysis. On the other hand, we assume that power consumption p_i is equal to the square of c_i [16] divided by t_i and is expressed as

$$p_i = \frac{c_i^2}{t_i}. \tag{2}$$

In this paper, we consider an offloading policy to offload the POW tasks of miners in the wireless mobile edge computing network. Given number of mobile users I, computing ability c_i, power consumption p_i, mining reward r_i, time delay t_i for locally mining a block of the i-th mobile user, the problem aims to generate a task offloading schedule for assigning maximum tasks to the edge computing unit with minimum power consumption and time delay on block generation. Therefore, we propose BDO (Block Data Offloading) to determine the optimal offloading schedule for mobile users so as to minimize the power consumption and delay on block generation.

3.2.2 Neural network architecture

Recently, the existing neural network architectures have been able to provide optimal solutions for various applications and societies [17]. Instead of designing a new neural network architecture, our major contribution focuses on building an existing neural network architecture based on an analytical model. In addition, our contribution is to devise the data training for the neural networks by using the valuations of the offloading utility for the mobile users. We design a neural network architecture by inputting all offloading demands from the users and outputting the

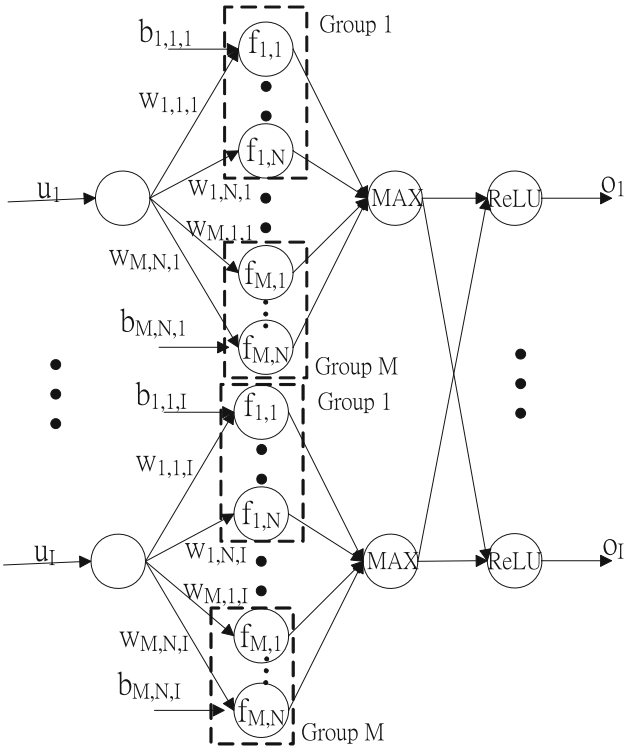

Fig. 5 Neural network architecture

optimal block offloading scheduling. Figure 5 shows the neural network architecture with multiple layers to perform the optimal block offloading scheduling. Specifically, each utility u_i is inputted to a two-layer feed forward network as shown in Fig. 5 . In the first layer, we obtain $f_i = w_i u_i + b_i$, where w_i and b_i are the weights and biases respectively, and i is from 1 to I.

In the second layer, we model the optimal data schedule on the edge computing unit. We schedule the block data with the probability of each offloading demand. Specifically, the output of the second layer is a vector of scheduling probabilities $\{f_i\}$. The maximum of f_i has high priority to generate a block by the edge computing unit. Therefore, we adopt a *ReLU* activation expressed as

$$g(f_i) = ReLU(f_i), \tag{3}$$

where i is from 1 to I, and *ReLU* function can search for the maximum of f_i to ensure a non-negative probability.

3.2.3 Training method

We design a training method to optimize weights and biases of a neural network by minimizing a loss function. We first define the linear function of a hidden layer in the m-th group of the n-th neuron for i-th user as $f_{m,n}(u_i) = w_{m,n,i}u_i + b_{m,n,i}$, where $1 \le m \le M, 1 \le n \le N$, and $1 \le i \le I$. Note that M is defined as the total number of groups on the neuron in the linear function of a hidden layer. A group of neurons are shown at dashed boxes in Fig. 5. *MAX* is defined as a max operation to maximize linear function $f_{m,n}(u_i)$ of a hidden layer in the m-th group of the n-th neuron for i-th user. Therefore, the network includes a layer of linear functions, a layer of max operations, and a layer of *ReLU* activation functions as shown in Fig. 5.

The training data are composed of utility descriptions of the mobile users. Specifically, the utility descriptions of the mobile users are generated by modeling independently and identically distributed functions. We denote the offloading demand descriptions of the i-th mobile user as $d_{i,j}$, where i is from 1 to I and j is the size of the training data. $d_{i,j}$ can be derived by t_i, p_i, r_i, c_i mentioned in Eq (1) such that $d_{i,j} = \frac{t_i \times p_i \times r_i}{c_i}$. The distribution $f_D(d)$ represents the distribution of $d_{i,j}$. The distribution of t_i is denoted by $f_T(t)$. The distribution of p_i is denoted by $f_P(p)$. The distribution of r_i is denoted by $f_R(r)$. The distribution of c_i is denoted by $f_C(c)$. The distribution f_D can be implied from distributions f_C, f_T, f_P, and f_R. Without loss of generality, we assume that c_i, p_i, r_i, t_i are independent uniform distributions from $[c_{\min}, c_{\max}]$, $[p_{\min}, p_{\max}]$, $[r_{\min}, r_{\max}]$, $[t_{\min}, t_{\max}]$ respectively [18, 19].

In terms of Eqs. (1) and (2), we can uniquely obtain $c_i = \frac{d_i}{x_i}$, $p_i = \frac{d_i^2}{t_i x_i^2}$, and $r_i = x_i$ for each i=1, 2, ..., I. That is, c, p, r can be derived in terms of x, t, d. Therefore, there exists an inverse transformation such that the transformation is an one-to-one map from C, P, R to X, T, D. We adopt a Jacobian matrix to transform the probability density function of c, p, r to the probability density function of x, t, d. The joint Probability Density Function (PDF) of x, t, d is

$$f_{XTD}(x,t,d) = f_{CPR}(c,p,r)|J|, \tag{4}$$

where Jacobian J of c, p, r with respect to x, t, d is given by

$$\begin{bmatrix} \frac{\partial c}{\partial x} & \frac{\partial c}{\partial t} & \frac{\partial c}{\partial d} \\ \frac{\partial p}{\partial x} & \frac{\partial p}{\partial t} & \frac{\partial p}{\partial d} \\ \frac{\partial r}{\partial x} & \frac{\partial r}{\partial t} & \frac{\partial r}{\partial d} \end{bmatrix} = \begin{bmatrix} \frac{-d}{x^2} & 0 & x^{-1} \\ \frac{-2d^2}{tx^3} & \frac{-d^2}{rt^2} & \frac{2d}{tr^2} \\ 1 & 0 & \frac{1}{c} \end{bmatrix} = \frac{2d^2}{t^2x^2}. \tag{5}$$

To obtain the marginal probability density function of d, we integrate the joint probability density function of x, t, d over x and t . The distribution of d is derived by

$$\begin{aligned} f_D(d) &= \int_{t_{\min}}^{t_{\max}} \int_{r_{\min}}^{r_{\max}} f_{CPR}(c,p,r)|J|dxdt \\ &= \frac{(t_{\max} - t_{\min})d^2}{2(c_{\max} - c_{\min})(p_{\max} - p_{\min})} \times \\ &\quad \frac{1}{t_{\min} t_{\max} r_{\min} r_{\max}}, \end{aligned} \tag{6}$$

where $d \in [\frac{t_{\min} \times p_{\min} \times r_{\min}}{c_{\max}}, \frac{t_{\max} \times p_{\max} \times r_{\max}}{c_{\min}}]$.

We define a loss function as

$$l(W,B) = -\sum_{i=1}^{N} (o_i(W,B) - f_D(d_i))^2, \tag{7}$$

where W and B are the weight matrix and the bias matrix contained with elements $w_{m,n,i}$ and $b_{m,n,i}$, respectively. o_i is the output of the neural network. We recursively minimize the loss function to search for the optimal W and B expressed as

$$\vartheta(W,B) = \arg\min_{W,B} \frac{1}{M} \sum_{p=1}^{M} l_p(W,B), \tag{8}$$

where M is the size of mini-batch. This is because the computational complexity with the mini-batch can be further reduced as compared to training whole datasets.

We use Stochastic Gradient Descent (SGD) [20] to update the weights and biases in the neural network. Let W_t and B_t be the t -th updates of W and B respectively in the training. SGD updates W_t and B_t respectively expressed by

$$W_t = W_{t-1} - \frac{\eta_t}{M}\sum_{p=1}^{M}\frac{\partial l_p(W,B)}{\partial W_{t-1}}, \tag{9}$$

$$B_t = B_{t-1} - \frac{\eta_t}{M}\sum_{p=1}^{M}\frac{\partial l_p(W,B)}{\partial B_{t-1}}, \tag{10}$$

where η_t is the learning rate for the t-th update. To adaptively adjust the learning rate for each update, we use Adagrad [21] to obtain adaptive learning rates as $\eta_t = \frac{\eta}{\sqrt{\sum_{k=1}^{t}(\frac{\partial l}{\partial W_k})^2}}$, where η is an initial learning rate.

3.2.4 Dropout algorithm

The average predictions of all different models are the most important since this can mitigate over dependence on some specific data features during the training of the neural networks. Therefore, dropout [22] is to randomly drop out hidden and visible units and input features during the training of fully connected neural networks. Such a way prevents the units from co-adapting and overfitting and improves the network's generalization ability and test performance. Traditionally, we keep the units of the m-th group on the hidden layer with probability p, otherwise set them to 0 with probability 1-p, where p is confined to Bernoulli distribution. However, a repeated sampling random subset of the dataset leads to huge time consumption in convergence. This is because it approximates optimum based on Monte Carlo optimization with the huge number of trials. Therefore, we adopt a Gaussian approximation to sample the random subset of the dataset in training.

We use a central limit theorem to approximate the Gaussian distribution of dropout. When the dropout is applied to the outputs of a fully connected layer, the output of the neural network times the probability of Bernoulli distribution expressed as

$$ReLU(w_{m,n,i}u_i + b_{m,n,i}) \times p_{m,n,i}, \tag{11}$$

where $1 \le m \le M, 1 \le n \le N$, and $1 \le i \le I$. Based on Eq. (11), we simplify the dropout output of the neural network with matrix formulation $o = \sum_{m=1}^{M} ReLU(W_m u_i + B_m)P_m$, where P_m is a binary mask matrix of the m-th group of neurons with each element drawn independently from Bernoulli. We observe that o is a weighted sum of Bernoulli random variables. In light of the central limit theorem, when $M \to \infty$, $\frac{o-M\mu}{\sqrt{M}\sigma}$ tends to a normal distribution, where μ is the mean of o and σ is the variance of o. In fact, such an approximation performs well under low dimensional data with $M = 15$. We empirically verify that the approximation is good for typical datasets of moderate dimensions, except when several dimensions dominate all others.

4 Experimental results

In this section, we first present the experimental setup on the micro-blockchain and task offloading algorithms in the mobile network. Then, we conduct the numerical simulations to evaluate the performance of our proposed micro-blockchain and task offloading methods in the mobile network.

4.1 Micro-blockchain performance analysis

4.1.1 Scenario setup

In this section, we present the experimental setup for various performance aspects in the micro-blockchain. We first implement the micro-blockchain based on an open source of mini-blockchain [6]. We use NS3 [23] to evaluate the performance of the micro-blockchain in the mobile social network since it has been a popular simulator for analyzing peer-to-peer networks. The NS3 simulator uses LTE modules with MEC Couldlets [24]. Note that the cloudlet-based MEC [25] can improve real-time mobile computing and reduces traffic burden of backbone networks. We setup 20 edge computing units and 50 mobile users for each unit. Each mobile user moves in a square area of $1Km \times 1Km$. We assume N to be 20. We adopt a community-based mobility model [26] with uniform distribution from 0 to 1 in attraction. The average of simulation runs is about 50, and the average of simulation time is almost 700 seconds. We set 500 users to generate 1000 transactions per second with a normal random variable under the mean of 900 transactions per second and the variance of 100 transactions per second. The arrival time of each mobile user follows a Poisson distribution.

In this section, we evaluate the performance of the proposed method with Bitcoin [1], MBC (mini-blockchain) [6], and LSB [4]. Note that the MBC includes a proof chain and records balances of all non-empty addresses in the account tree while it still adopts POW to verify the blocks. On the other hand, LSB [4] presents a light-weight blockchain for IoT and introduces a simple block verification to replace POW.

4.1.2 Evaluation of computational cost

In this section, we evaluate the average time consumption of each consensus algorithm for generating a block under 100 mobile users as shown in Fig. 6. Bitcoin results in the worst time consumption of POW about 1424.48 seconds since it requires the SHA-256 hash of the block to search for the correct nonce where the size of the block is set to 500kB. The POW-based MBC also leads to almost 100.81

seconds to generate a block. On the other hand, LSB performs fair as 5.31 seconds. Our method is superior to other methods with the consensus time as 2.47 seconds since the root hash and the header hash merely need several hash functions. Figure 6 shows that traditional POW incurs significant delays to burden the mobile network.

4.1.3 Evaluation of scalability

In this section, we investigate the average processing time of transactions by increasing the total number of mobile users as shown in Fig. 7. Increasing the number of the mobile users causes the increase of pending transactions. Bitcoin that broadcasts all blocks to all mobile nodes leads to the worst propagation delay due to huge broadcast flooding. The average processing time of Bitcoin is extremely huge in large scale. MBC is better than Bitcoin since it can reduce the size of blocks in the chain and the overhead of block broadcast in large scale. Since LSB distributes the workload of block verification to head nodes, it can give fair scalability. Our method performs very well since each edge computing unit offloads the transaction process for each mobile social community.

4.1.4 Evaluation of throughput

In this section, we measure the throughput of each method when the network workload fluctuates as shown in Fig. 8. Since LSB can dynamically adjust the network utilization based on the number of generated transactions, LSB yields better results. However, LSB still needs head nodes to compete against each other for generating new block and thereby results in lower throughput than our proposed method. The reason is that the proposed method can achieve high throughput by using the mobile edge computing unit to verify the transactions and blocks. On the other hand, Bitcoin has fixed throughput with a fixed throughput of 7 transactions per second. Although MBC prunes the traditional blockchain, the POW manner of MBC gives fair performance.

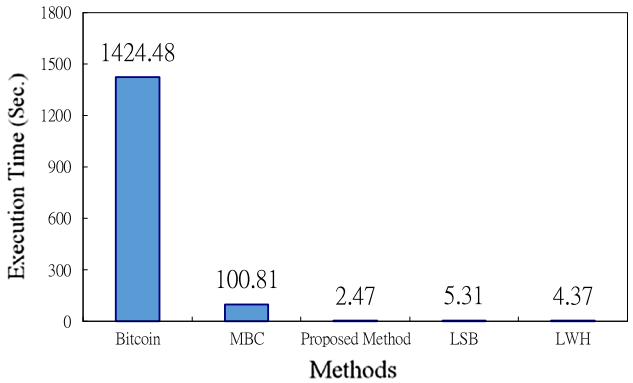

Fig. 6 Execution time of each method

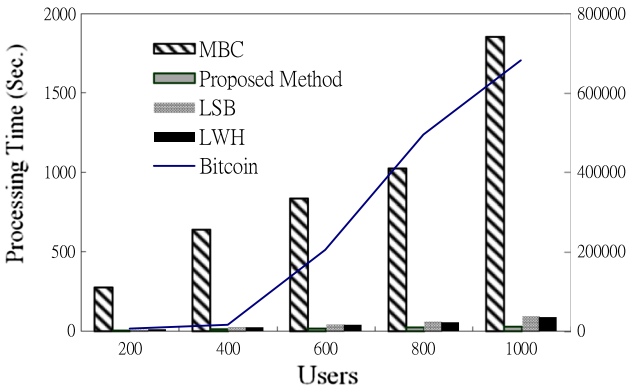

Fig. 7 Processing time of each blockchain method under the different number of users

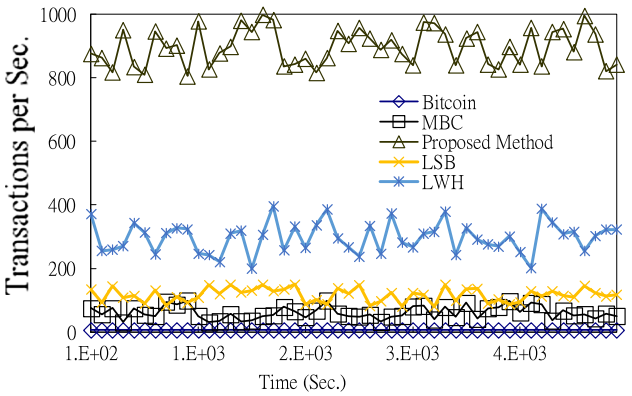

Fig. 8 Throughput of each blockchain method under different time slots

4.1.5 Discussion

We further compare the proposed micro-blockchain with a previous blockchain structure, named LWH (light-weight hash) [27] based on an assumption of the same proposed offloading scheme. LWH simplifies the hash functions of blockchain for the resource-constrained IOT scenarios. Figures 6, 7 and 8 show that our proposed method exhibits shorter processing time and higher throughput than LWH. This is because the proposed micro-blockchain has a concise account tree to efficiently and safely record transactions instead of computing complex hash functions and cryptographic calculations. Although LWH derives a simplified hash function in the blockchain, generic mobile devices still cannot afford the inevitable complex computation in the hash function.

4.2 Task offloading performance analysis

4.2.1 Scenario setup

We compare our method to task offloading based on the deep learning without dropout (No DO) and the task offloading based on the deep learning with Bernoulli-based Monte Carlo dropout (MC). In Bernoulli dropout, we follow the dropout parameter recommendations [22] [28] to set the dropout rate of $p = 0.5$ for the hidden layers and $p = 0.2$ for the input layer. We conduct the training data by using the TensorFlow [29] deep learning library. We set the training set as 100000 offloading demand profiles and sample t_i, p_i, r_i, c_i from $f_T(t)$, $f_P(p)$, $f_R(r)$, $f_C(c)$, respectively. t is set as a uniform distribution from 0.1 to 0.3. p is set as a uniform distribution from 0.2 to 0.5. r is set as a uniform distribution from 0.1 to 0.7. Distribution of c is set as a uniform distribution from 0.1 to 0.6. We train the neural network for 600 epochs, where the batch size is 800. Initial learning rate η is set as 0.01. The number of mobile users is from 1000 to 100000. We then calculate the gradients of the weights on the trained set.

4.2.2 Performance analysis

We first explain the optimal aspects for the experimental investigation between the proposed method and LBTO (Location Based Task Offloading) [30]. LBTO is a location-based task offloading scheme, which can make offloading decisions based on high speed mobility on end users. On the other hand, our proposed method adopts a deep learning approach to fast determine an optimal schedule of task offloading on the edge computing unit under the constraints of time delay and power consumption. Therefore, the utilities of offloading are maximized to improve the throughput in the proposed micro-blockchain. However, traditional non-optimal works such as LBTO do not jointly consider the variety of computational capabilities in each mobile user and a limitation of battery on end users to perform the optimal task offloading. Additionally, the conventional algorithms cannot immediately solve such NP-hard combinatorial optimization problems within time constraints. This is because they cannot yield a real-time offloading policy for the fast variety of statuses in MEC such as the computational ability, user mobility, and the battery limitation on the end users. Instead of solving the time-consuming problem, the proposed method with a deep neural network can quickly give an optimal solution by learning the offloading policy from the training. Such a way can simplify the problem solvable by a deep neural network with the complexity $O(N)$, where N is the number of users performing the offloading. However, the conventional numerical optimization methods such as LBTO take lots of iterations to solve the problem with the complexity $O(N^2)$.

We conduct experiments to evaluate the performance between the proposed method and LBTO in Figs. 9, 10 and 11. Figures 9, 10 and 11 show that our proposed method outperforms LBTO. LBTO chooses edge computing units only considering the best transmission paths from the mobile users to the edge computing units. However, other ignored factors such as the battery of the mobile user seriously affect the transmission paths in the offloading efficiency. The reason is that the status of the battery in the mobile user dominates the distance of the best transmission paths. Therefore, our proposed method is superior to LBTO since our scheme jointly considers both computation resource as well as the user states to offload tasks for the mobile users in the varying MEC scenarios.

We then discuss the differences between the proposed method and other general throughtput adjustments. RANDOM (Random throughput adjustment) [31] is a classic algorithm of generic throughput adjustments. RANDOM randomly assigns tasks of users to the edge computing unit. However, RANDOM cannot achieve the efficient offloading since it just randomly adjusts the throughput based on the assumptions of the unlimited battery and the constant computing capabilities on the end users. These assumptions are unsatisfied with the real wireless mobile network. Therefore, Fig. 11 shows that our proposed method overpasses RANDOM since the dynamic throughput adjustment can accommodate highly dynamic wireless mobile network scenarios. The proposed deep learning method can real-time offload tasks with considering both the constraint of time delay and the power consumption on the end users.

Figures 12 and 13 show that the proposed method can induce a faster convergence to the solution compared to

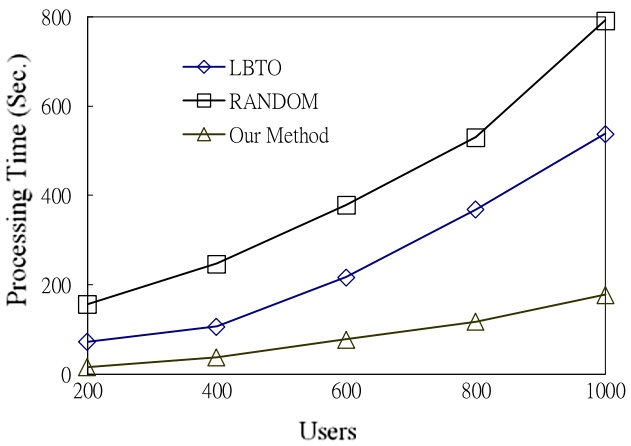

Fig. 9 Processing time of each offloading method under the different number of users

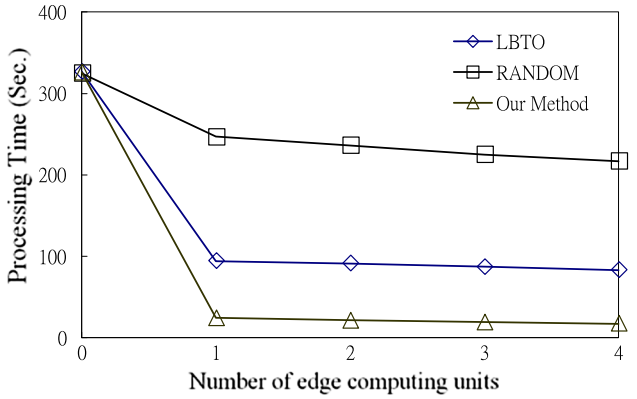

Fig. 10 Processing time of each offloading method under the different number of edge computing units

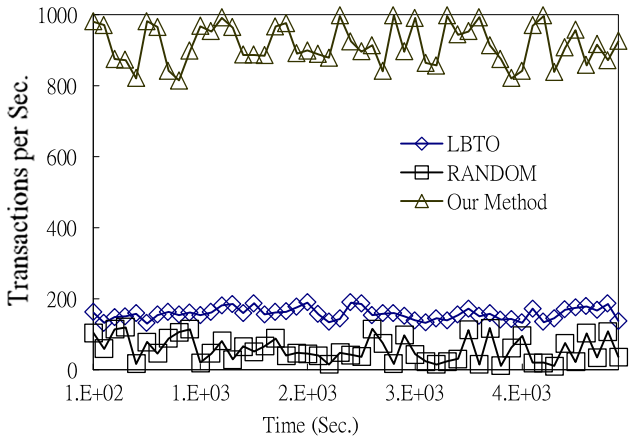

Fig. 11 Throughput of each offloading method under different time slots

other methods. The proposed method with the approximating Gaussian dropout is about 8 times faster than the traditional MC dropout. Although the proposed method is slightly slower than No DO, the proposed method can achieve higher performance to reduce the average execution cost. The proposed method can be converged within 92 epochs as shown in Fig. 12.

5 Conclusion

Blockchain is a technique to realize cryptocurrency and M-commerce convenience in a social mobile computing environment. However, traditional blockchain without considering scalability leads to large power consumption and huge computational cost in the mobile social network. In this paper, the problem of devising a scalable blockchain for the mobile social network was addressed. In view of the characteristics of the mobile social network, we proposed an algorithm to generate a light-weight blockchain to avoid the above drawback so as to minimize the transaction time

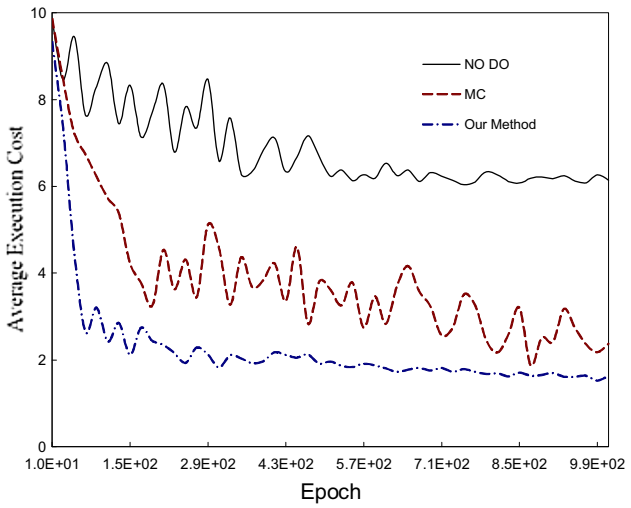

Fig. 12 Average execution cost of each dropout method under different epochs

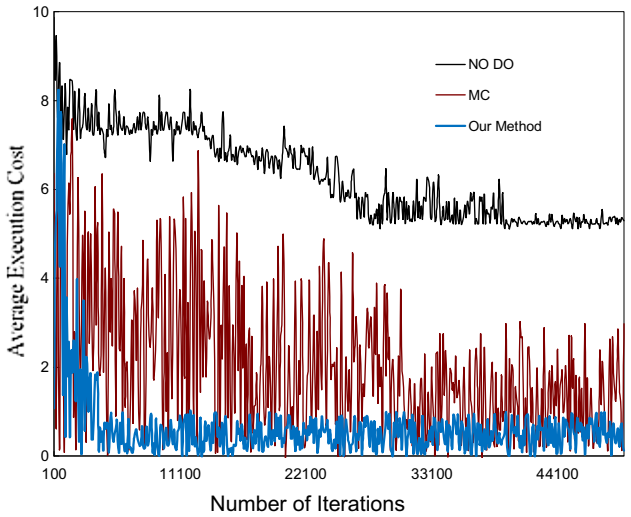

Fig. 13 Average execution cost of each dropout method under the different number of iterations

of users. We also proposed methoYd to fast process consensus such that our proposed method can achieve higher throughput as compared to traditional credit cards or mobile payments. In addition, we adopted a mobile edge computing to offload block verification for mobile users. We devised an offloading algorithm based on deep learning to schedule the tasks of the block verification for each mobile user. Therefore, the average execution cost of the block verification was minimized in the mobile edge computing scenario. In order to evaluate the performance of the proposed method, we have conducted several empirical experiments. During the experiments, we analyzed the effectiveness of the proposed method in terms of the average transaction time. We also inspected the efficiency of the proposed method by measuring its

computational cost. Experimental results have showed that the proposed method is of very high scalability and practically useful in the mobile social network.

References

1. Nakamoto, S. (2008). Bitcoin a peer-to-peer electronic cash system. Retrieved from http://www.bitcoin.org/bitcoin.pdf
2. Rausch, P., Hashemi, S. H., Faghri, F., & Campbell, R. H. (2016). World of empowered IoT users. In *IEEE first international conference on internet of things design and implementation (IoTDI)*.
3. Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *IEEE Access*, *4*, 2292–2303.
4. Jurdak, R., Dorri, A., Kanhere, S. S., & Gauravaram, P. (2019). Lsb a lightweight scalable blockchain for iot security and privacy. *Journal of Parallel and Distributed Computing.*, *134*, 180–197.
5. Niyato, D., Wang, P., Xiong, Z., Feng, S., Han, Z. (2017). Edge computing resource management and pricing for mobile blockchain. arXiv preprint.
6. Bruce, J. (2014). The mini-blockchain scheme. www.cryptonite.info.
7. Pouwelse, J., Otte, P., de Vos, M. (2017). Trustchain: A sybil-resistant scalable blockchain. In *Future generation computer systems*. Elsevier.
8. Samet, H. (1989). *The design and analysis of spatial data structures*. Boston: Addison-Wesley.
9. Bayer, S. H. D., & Stornetta, W. S. (1993). *Improving the efficiency and reliability of digital time-stamping* (pp. 329–334). New York: Springer.
10. Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. vol. 151. Ethereum Project Yellow Paper.
11. Aitzhan, N. Z., & Svetinovic, D. (2016). Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing.*, *15*(5), 840–852.
12. Stanciu, A. (2017). Blockchain based distributed control system for edge computing. In *International conference on control systems and computer science (CSCS)* (pp. 667–671).
13. Samaniego, M., & Deters, R. (2016). Blockchain as a service for iot. In *IEEE international conference on internet of things (iThings)* (pp. 433–436).
14. Omohundro, S. (2014). Autonomous technology and the greater human good. *Journal of Experimental and Theoretical Artificial Intelligence*, *26*(3), 303–315.
15. Tao, X., Ota, K., Dong, M., Qi, H., & Li, K. (2017). Performance guaranteed computation offloading for mobile-edge cloud computing. *IEEE Wireless Communications Letters*, *6*, 774–777.
16. Zhang, J., Mao, Y., & Letaief, K. B. (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, *34*(12), 3590–3605.
17. Liu, X., Zeng, N., Liu, Y., Alsaadi, F. E., Liu, W., & Wang, Z. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*, 11–26.
18. Gerding, E., Perez-Diaz, A., & McGroarty, F. (2018). Decentralised coordination of electric vehicle aggregators. In *International workshop on optimization in multiagent systems*.
19. Kilari, V. T., Yang, D., Tang, J., Yu, R., & Xue, G. (2018). Coinexpress: A fast payment routing mechanism in blockchain-based payment channel networks. In *International conference on computer communication and networks (ICCCN)*.
20. Monga, R., Chen, K., Devin, M., Dean, J., Le, Q. V., Mao, M., Corrado, G., Ranzato, M. A., Senior, A., Tucker, P., Yang, K. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems (NIPS)*. (Vol. 1, pp. 1223–1232)
21. Hazan, E., Duchi, J., & Singer, Y. (2011). Adaptive subgradient methods for online leaning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.
22. Krizhevsky, A., Sutskever, I., Hinton, G. E., Srivastava, N., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. CoRR, abs1207.0580.
23. Ns3. https://www.nsnam.org (2017).
24. Github. (2016). Network simulator for edge computing and cloud computing. https://github.com/subinjp/edgecomputing.
25. Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, *8*(4), 14–23.
26. Harri, J., Filali, F., & Bonnet, C. (2009). Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE Communications Surveys & Tutorials*, *11*(4), 19–41.
27. Park, J., Seok, B., & Park, J. H. (2019). A lightweight hash-based blockchain architecture for industrial Iot. *Applied Sciences*, *9*(18), 3740.
28. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, N. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.
29. TensorFlow. (2017). https://www.tensorflow.org/.
30. Chen, X., Zhong, W., Yang, C., Liu, Y., & Xie, S. (2019). Efficient mobility-aware task offloading for vehicular edge computing networks. *IEEE Access*, *7*, 26652–26664.
31. Chen, M., & Hao, Y. (2018). Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications*, *36*(3), 587–597.

Chung-Hua Chu received B.S. from National Kaoshiung First University of Science and Technology, Department of Computer and Communication, Kaoshiung,Taiwan, in 2002. He joined the Multimedia and Communication Lab with advisor Prof. Yen-Chieh Ouyang at National Chung Hsing University from 2002 to 2004. He received M.S. from National Chung Hsing University,Taichung,Taiwan, in 2004. He joined the Network Database Lab with advisor Prof. Ming-Syan Chen at National Taiwan University from 2004 to 2008. He received Ph. D. from National Taiwan University, Graduate Institute of Communication Engineering in 2008. He served military services in 2008. He has been an assistant Professor at Department of Multimedia Design, National Taichung University of Science and Technology, Taiwan, since 2009.