Intelligent Offloading in Blockchain-Based Mobile Crowdsensing Using Deep Reinforcement Learning

Zheyi Chen[†,*] and Zhengxin Yu[§]
[†]College of Computer and Data Science, Fuzhou University, Email: z.chen@fzu.edu.cn
[§]School of Computing and Communications, Lancaster University, Email: z.yu8@lancaster.ac.uk
[*]Corresponding author

***Abstract*—Mobile Crowdsensing (MCS) utilizes sensing data collected from users' mobile devices (MDs) to provide high-quality and personalized services, such as traffic monitoring, weather prediction, and service recommendation. In return, users who participate in crowdsensing (i.e., MCS participants) get payment from cloud service providers (CSPs) according to the quality of their shared data. Therefore, it is vital to guarantee the security of payment transactions between MCS participants and CSPs. As a distributed ledger, the blockchain technology is effective in providing secure transactions among users without a trusted third party, which has found many promising applications such as virtual currency and smart contract. In a blockchain, the proof-of-work (PoW) executed by users plays an essential role in solving consensus issues. However, the complexity of PoW severely obstructs the application of blockchain in MCS due to the limited computational capacity of MDs. To solve this issue, we propose a new framework based on Deep Reinforcement Learning (DRL) for offloading computation-intensive tasks of PoW to edge servers in a blockchain-based MCS system. The proposed framework can be used to obtain the optimal offloading policy for PoW tasks under the complex and dynamic MCS environment. Simulation results demonstrate that our method can achieve a lower weighted cost of latency and power consumption compared to benchmark methods.**

I. Introduction

The vigorous development of mobile devices (MDs) with built-in sensors enables Mobile Crowdsensing (MCS) to be a flexible and low-cost pattern for collecting sensing data, which promises an efficient service provisioning for the Internet-of-Things (IoT) applications [1]. Fig. 1 shows a typical architecture of an MCS system. First, IoT applications publish sensing tasks to cloud service providers (CSPs), which design and release the payment strategy accordingly. Next, users who participate in crowdsensing (i.e., MCS participants) execute the sensing tasks by using their MDs and upload the sensing data to CSPs. After CSPs evaluate the quality of received data, MCS participants can receive their deserved payment. Finally, the sensing data is uploaded and utilized by different IoT applications, such as traffic monitoring, weather prediction, and service recommendation.

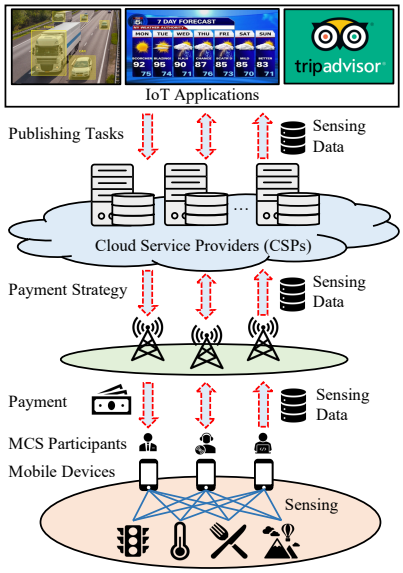

Fig. 1. The architecture of a Mobile Crowdsensing (MCS) system.

In an MCS system, a reliable trading mechanism needs to be established as the security guarantee for payment transactions between MCS participants and CSPs. However, most of the traditional transaction-management systems validate and store payment transactions in a centralized manner, which may pose severe security risks and performance bottleneck at a central agency. Such a centralized manner is fragile to the single-point failure and thus cannot offer reliable services, which is unacceptable in modern MCS systems. By contrast, as a distributed and append-only ledger, blockchain provides a trustworthy solution for transactions on the Bitcoin network [2]. Each transaction must be verified (e.g., using the digital signature) by all users, who run the proof-of-work (PoW) [3] for reaching the global consensus before a transaction is appended to the blockchain on the Bitcoin network. As a result, the conflicting transactions from malicious users would be automatically denied and dropped. Therefore, the blockchain technology can be used to secure trans-

actions in the MCS system. Moreover, when a user successfully adds a new block to the blockchain, it will receive a certain reward in return. However, due to the limited computational capacity of MDs, the application of blockchain in MCS is severely hindered by the computation-intensive tasks of PoW during the mining process of blockchain.

To address the above challenge, one promising approach is to offload the computation-intensive tasks of PoW from MDs to the nearby edge servers, which are more powerful and capable of reducing the cost of latency and power consumption [4]. However, most of classic offloading strategies are based on heuristics [5] and game theory [6], which cannot fully adapt to changeable MCS environments. To tackle this problem, Reinforcement Learning (RL) has emerged as an effective strategy [7], learning the optimal offloading policy through interacting with the environment.

The current RL-based approaches may handle the offloading problem to some extent [8], [9], [10], but the majority of them discretize continuous values (e.g., latency or power consumption) via utilizing the value-based RL (e.g., Q-learning and Deep Q-Networks (DQN)). Thus, the integrity of continuous space is lost and meanwhile the noise is also created, making it hard to obtain the optimal offloading policy. In contrast, the policy-based RL (e.g., policy gradient) is better at handling the offloading problem in a large space. However, the policy-based RL may cause high variance when calculating the gradient, resulting in low training efficiency. In order to address these important challenges, we design a new Deep Reinforcement Learning (DRL) based framework for effectively offloading the computation-intensive tasks of PoW to edge servers in a blockchain-based MCS system. With the help of DRL, the proposed framework can be used to obtain the optimal offloading policy for PoW tasks under the complex and dynamic MCS environment. The main contributions of this article are summarized as follows.

- A blockchain-based framework is developed to effectively secure the transactions between MDs and CSPs in an MCS system, where PoW is used to guarantee the global consensus and thus transactions are consistent, unique and unfalsified. The difficulty of PoW increases with the growing number of MDs, in order to avoid conflicts among MDs when they try to add blocks into the blockchain simultaneously.
- A new DRL-based framework is proposed to offload the computation-intensive tasks of PoW to edge servers, where a Markov Decision Process (MDP) is used to build the system model. Notably, by integrating Proximal Policy Optimization (PPO) and Differentiable Neural Computer (DNC), which is able to access and perform differentiable read-write operations on structured external memories in an objective-oriented manner, and thus the decision-making efficiency is significantly improved.
- Simulation results verify the effectiveness of the proposed intelligent offloading framework, which achieves a lower weighted cost of latency and power consumption compared to benchmark methods.

The rest of this article is organized as follows. The next section introduces the blockchain-based MCS for secure transactions. The proposed offloading framework for a blockchain-based MCS system is then described. Next, an intelligent offloading method is presented. Finally, we conduct the performance evaluation and conclude the article.

II. Blockchain for Secure Transactions in MCS

Bitcoin's basic concept was suggested in 2008 [3], which has subsequently grown in popularity as decentralized digital currency. A peer-to-peer (P2P) network underpins Bitcoin, where exists the consensus issues of synchronization, falsification, and reuse. On the Bitcoin network, blockchain [3] was designed to tackle the above consensus issues, which is used as a distributed ledger maintained by all Bitcoin users. Specifically, a blockchain is made up of numerous blocks stored in a single chain that has been verified by the majority of users. A transaction is placed into a new block and this block will be added to the blockchain after the PoW task is completed with verification from the majority of users. Therefore, the blockchain offers a reliable mechanism for ensuring secure transactions and preventing transactions from the attacks of malicious users.

In order to ensure the security of payment transactions in MCS, a blockchain-based framework is designed for the MCS system. As illustrated in Fig. 2, an MD first adds a transaction with CSPs to a new block being made. After completing the new block,

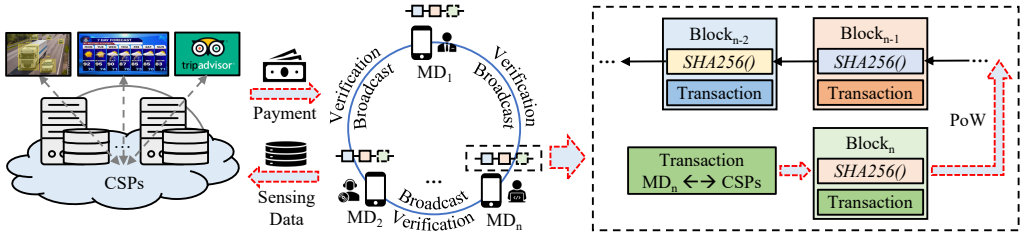

Fig. 2. The blockchain-based architecture for secure transactions in an MCS system.

the MD broadcasts this new block to other MDs on the blockchain network. Next, each of the other MDs receives the new block and verifies it (e.g., using the digital signature) for security and data integrity. If the new block passes the verification, the MDs will add it to the blockchain they maintain. Otherwise, this new block will be dropped.

Different new blocks may be generated by different MDs at the same time, but only one blockchain is recognized and maintained by the MDs. Therefore, the MDs must select the same new block and add it to the existing blockchain. However, it is difficult to develop a uniform selection criterion due to the equality and non-priority among different blocks. Inspired by the blockchain technology, we restrict the number of new blocks generated by MDs in unit time. More specifically, after an MD fills a transaction into a new block, some extra work needs to be completed before broadcasting the new block on the blockchain network. The process of this extra work can be divided into two steps as follows.

Step 1: The contents of a block are integrated into a string, which consists of the $SHA256()$ value of the preceding block, the basic information of the current block, and the transaction in the current block. Each block has a one-to-one correspondence with its own $SHA256()$ value, which is a cryptographic hash function, to represent the block [11]. As shown in Fig. 2, every block contains the $SHA256()$ value of the preceding block. Thus, the blockchain relies on the $SHA256()$ value of each block to connect all blocks orderly.

Step 2: A random number is added to the content string of a block generated in Step 1 to form a new string. Next, this new string is input into $SHA256()$ to calculate a binary number of 256 bits. When the generated binary number is smaller than the set $target$ (representing the mining difficulty), this extra work is regarded to be completed. Especially, we propose that the value of $target$ inclines with the growing number of MDs in order to avoid the conflicts among MDs when they try to add their blocks into the blockchain simultaneously. In this way, the blockchain-based framework can achieve high reliability under complex network scenarios with many MDs.

This extra work is known as PoW in the mining process of blockchain, which has been established as a mathematical conundrum that is exceedingly hard to resolve yet simple to verify. Therefore, it requires massive computational resources to complete the computation-intensive tasks of PoW. Furthermore, to verify the effectiveness of the proposed framework, a classic threat model (i.e., collusion attack) in MCS is considered. In the phase of the transaction, some MCS participants collude with CSPs. Through offering the communication information of other participants, CSPs reveal the background knowledge and privacy information of other participants, which forms a malicious interaction between MCS participants and CSPs.

However, due to the limited computational capacity of MDs, it is challenging to apply the blockchain technology into wireless mobile networks for secure transactions in an MCS system. To address this problem, we first design a new credit-based incentive mechanism, where the user credit is used to measure the contributions of each participant to a sensing task published in MCS systems. In general, the user credit has a positive correlation with the volume of valid data uploaded by each participant. If an MCS participant makes more contributions to a sensing task in MCS systems (i.e., uploads more valid sensing data), it will receive higher credit and thus its mining difficulty can be reduced to some extent. On the contrary, it can only obtain fewer advantages when dealing with PoW tasks. By using the proposed credit-based incentive mechanism, MCS participants with higher credit can save more computational resources for processing PoW tasks. Conversely, participants with lower credit may spend more computational resources to work out PoW. Moreover, participants with a higher credit can enjoy the priority of broadcasting and verifying

new blocks when multiple participants try to add blocks simultaneously since their PoW tasks are easier to solve. The above factors motivate participants to contribute to the sensing tasks published in MCS systems and the global consensus on the blockchain network. Next, to further relieve the limitation of computational capacity on MDs, the computation-intensive tasks of PoW is proposed to be offloaded to edge servers in a blockchain-based MCS system.

III. Efficient Computation Offloading in the Blockchain-Based MCS System

As shown in Fig. 3, a computation offloading framework is designed to reduce the cost (latency and power consumption) of PoW tasks processing in a blockchain-based MCS system. The proposed framework consists of multiple mobile edge clouds and MDs, where MDs offload tasks to the edge servers in the nearest mobile edge clouds through access points (APs). For the clarity of presentation, the scenario of a single mobile edge cloud is considered and the bandwidth of the wireless network is assumed to be evenly shared among MDs.

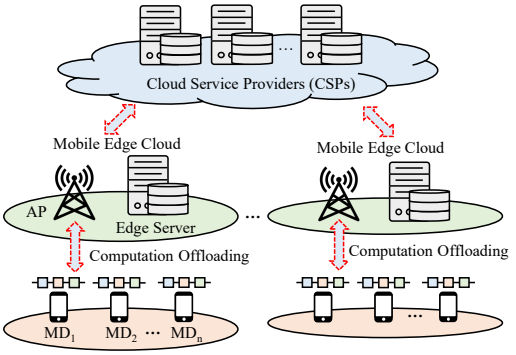

Fig. 3. Computation offloading in the blockchain-based MCS system.

After an MD completes a transaction in MCS, it will put the transaction into a block and try to add the block into the blockchain maintained in MCS for secure transactions. In order to add its block to the blockchain, the MD must first solve a PoW task, and then broadcast the block that is to be verified by other MDs. With our proposed offloading framework, the computation-intensive tasks of PoW can be offloaded to the edge servers. More specifically, we define a PoW task as a 2-tuple. The first element represents the size of input data (i.e., the contents of a block), which is regarded as the input of $SHA256()$. The second element indicates the required computational resources (CPU cycles) to complete the task, which is positively related to the number of hashes for calculating the qualified value of $SHA256()$ (one hash per CPU cycle). Besides, the computational capacity (Hz or CPU cycles/second) of edge servers and MDs are also defined in our proposed offloading framework.

Considering the case that all edge servers and MDs can provide computing services for processing PoW tasks, each PoW task can be executed locally or offloaded to the edge servers. Therefore, we define the following two modes.

Local Mode: When a PoW task is executed locally, we define the corresponding latency and power consumption of task processing, which make up the cost of local mode. This cost depends on the required computational resources and the computational capacity of MDs.

Edge Mode: When a PoW task is offloaded to an edge server for execution, the whole process can be described as follows, where we can obtain the cost of edge mode as a combination of latency and power consumption.

Step 1: An MD uploads a task to AP via the wireless network, which is then forwarded to an edge server. Therefore, we define the transmission latency, which considers the uplink rate of an MD in the wireless channel [12]. Moreover, the power consumption in this step is the product of transmission latency and transmission power.

Step 2: The edge server allocates computational resources and executes the task. The processing latency depends on the ratio of required and allocated computational resources to process the task. Moreover, the power consumption in this step is the product of processing latency and the total power, which is the sum of the processing power of edge servers and idle power of MDs.

Step 3: The MD downloads the result from the edge via AP, which is a random number that fulfills the $target$ of PoW. Since the data size of this number is very small, the latency and power consumption of downloading this result can be omitted.

As the latency and power consumptions have different value ranges, they are normalized into values of the same scale in our model. Finally, through integrating the cost of the above two modes, we can work out the total cost of our proposed offloading framework in a blockchain-based MCS system with different offloading decisions.

IV. DRL-Based Decision Making for Computation Offloading

Reinforcement Learning (RL) intends to learn an optimal policy through interacting with the environment [7]. In general, an RL agent chooses actions based on the observation of the current state, and the environment will feedback the corresponding reward and steps to the next state. In order to obtain an optimal policy that maps states to actions with maximized long-term reward, the RL agent will accordingly change the probability of choosing actions based on the current state and rewards received.

Normally, RL needs to record the value of each state-action pair in a table, which is feasible when the state space is with a low dimension. However, complex tasks in the real world often come with large state space. In this case, the traditional RL approach is unfeasible. To solve this issue, Deep Reinforcement Learning (DRL) [13] was developed to handle the high-dimensional state space with the help of deep neural networks (DNNs) [14], where DNNs can automatically find the low-dimensional representations of high-dimensional input. Through integrating RL and DNNs, DRL can directly learn an optimal policy from the high-dimensional input.

Therefore, we propose a DRL-based decision-making method to find the optimal offloading policy for a blockchain-based MCS system, in order to minimize the weighted cost of latency and power consumption. As shown in Fig. 4, a scenario of offloading in the blockchain-based MCS is deemed as the environment, and the DRL agent chooses actions via interacting with the environment. Next, the corresponding state space, action space and reward function are defined as follows.

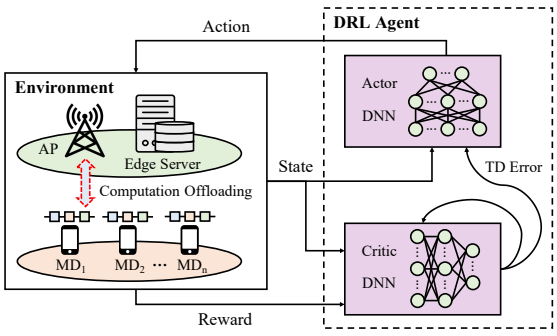

Fig. 4. The DRL-based framework for computation offloading.

State Space: It consists of three parts, which are the total cost of computation offloading, the available computational capacity of the edge server, and the network status (e.g., network band- width and uplink rate of each MD). Thus, the state is denoted as a 3-tuple, which takes into account both the system overhead and system resource.

Action Space: It is to make offloading decisions for the PoW tasks from MDs. Therefore, the action is defined as $\mathcal{A} = \{0, 1\}$ to represent the binary decision for offloading. If $\mathcal{A} = 1$, the PoW tasks will be processed locally. Otherwise, they will be offloaded and processed by the edge server.

Reward Function: It is used to guide the DRL agent to optimize the offloading policy with higher rewards, in order to minimize the total offloading cost. Therefore, the reward function is defined as the negative increment of the total cost.

State-transition model: It indicates the probabilities of transiting to the next state when taking actions at the current state. The state transition can also be regarded as the joint probability distribution of the next state and the corresponding rewards, where the transition frequency is used to estimate the transition probabilities from one state to others.

During the learning process, the DRL agent chooses an action at the current state. Next, the environment feedbacks a reward and steps to the next state. The above process is described as an MDP. However, it would be infeasible to use a precise mathematical model to handle the dynamic problem of computation offloading in the MCS environment. In response to the uncertainty of state in MCS, we use a model-free RL method that does not have to independently learn a fixed model for a specific MCS environment.

In our method, Proximal Policy Optimization (PPO) [15] is adopted to train DNNs for the optimal policy of offloading decisions.

Our proposed method is an actor-critic based hybrid RL algorithm, which combines the advantages of the classic value-based and policy-based RL algorithms. The classic value-based RL (i.e., critic-only) adopts the temporal-difference (TD) learning [7] to estimate the TD error and achieve low variance, where the ϵ-greedy is commonly used to make a balance between exploration and exploitation. However, the overhead of searching for the optimal action by using the ϵ-greedy is huge, especially when the space is continuous. In contrast, the classic policy-based RL (i.e., actor-only) parameterizes the policy and can directly optimize the problem with a continuous space. However, it conducts high variance when estimating gradient,

resulting in slow learning speed. To address these issues, our proposed method integrates the advantages of the above two RL algorithms, aiming to efficiently achieve low-variance results for the complex offloading problem. In our proposed method, the critic evaluates the action at the current state and decides the value function, which is used to update the gradient in the actor. Therefore, the actor can choose better actions referring to the critic's evaluation results of previous actions.

However, the PPO is commonly equipped with a simple design of replay memory, while too much redundant information will be introduced if all system states are stored. To address this problem, we combine PPO with DNC, aiming to improve its scalability and efficiency to find the optimal offloading policy. Specifically, the DNC utilizes a network controller to learn the way of reading and writing in a memory module, which enables the DRL agent to be trained in an objective-oriented manner. Thus, the proposed DRL-based method can better adapt to changeable blockchain-based MCS environments and achieve the optimal offloading policy more efficiently.

V. Performance Evaluation

We consider a mobile edge cloud consisting of one edge server and one AP [9], where the radius of the network coverage is 250 meters and the bandwidth is 20 Mbps. The computational capacity of an MD and edge server are 1 GHz (CPU Gigacycles/second) and 40 GHz (CPU Gigacycles/second), respectively. Meanwhile, the size of input data is limited to the maximum of a block (i.e., 1 MB) [3]. We simulate the proposed blockchain-based MCS system and the basic transaction process by using Python 3.6. The computational resources needed (in CPU Gigacycles) for completing PoW tasks are positively related to the number of hashes for calculating the value of $SHA256()$, which grows with the increase of the number of MDs for maintaining high reliability of the blockchain network. Moreover, the processing power of an edge server is 250 W, and the transmission and idle power of an MD are set to 1 W and 100 mW [9], respectively. As for the network design, we use two hidden layers that contain 200 and 100 neurons, respectively. Meanwhile, we set the number of training epochs as 1000, the reward decay rate as 0.95, the mini-batch size as 100, the actor's and critic's learning rate as 0.0001 and 0.0002, the TD error discount factor as 0.9, and the size of replay memory as 500. Also, we compare the proposed method with the other five schemes including the Local, Edge, Greedy, DQN, and PPO, where the network design of the DQN and PPO refers to the settings in the proposed method.

Table I shows the total cost (i.e., the weighted cost of latency and power consumption) of different offloading schemes with various numbers of MDs. Generally, the total cost rises with the increasing number of MDs. When the number of MDs is small, the Local has a poor effect. Because only local computational resources are utilized without the help of the edge server, which results in a high cost. When the number of MDs is large, the Edge performs poorly. Because only edge computational resources are utilized and the resources required to process PoW tasks have exceeded the computational capacity of the edge server. Although the performance of the Greedy always seems good, it may easily result in the local optimum because it only considers immediate rewards rather than long-term ones. Therefore, the Greedy cannot handle well the complex scenario with many MDs and the performance gap between the Greedy and the proposed method becomes larger as the number of MDs increases. By contrast, the proposed method works better than the other schemes while approximating the optimum with the increasing number of MDs. The optimum is found by trying all possibilities, which results in extremely-high complexity.

TABLE I
Comparison between baselines and the proposed method

No. of MDs	**Total cost of different offloading schemes** (10^{-3})				
	Optimum	*Local*	*Edge*	*Greedy*	*Proposed method*
20	6.302	12.798	7.991	7.087	6.523
40	30.907	42.275	52.796	32.705	31.346
60	65.707	74.982	107.901	68.769	65.786
80	101.463	112.312	163.504	109.232	102.084
100	136.941	152.284	227.675	148.830	137.293

Next, a performance comparison is made between the proposed method and the other two advanced DRL-based methods including the DQN and PPO. As shown in Table II, when the number of MDs is small, the DQN and PPO can achieve good performance that is slightly worse than but still comparable to the proposed method, which approximates to Optimum. However, when the number of MDs increases, the performance gap between these two methods and the Optimum becomes larger.

By contrast, the proposed method can effectively handle this problem in the scenario with many MDs and always approximate the Optimum. Due to the simple design of replay memory in the DQN and PPO, too much redundant information may be introduced and thus they cannot achieve high decision-making efficiency for the optimal policy. Through integrating the DNC, our proposed method is able to use a network controller to train the DRL agent in an objective-oriented manner and efficiently approach the optimal offloading policy. Therefore, the proposed method outperforms the other two DRL-based methods.

TABLE II
COMPARISON AMONG DIFFERENT DRL-BASED METHODS

No. of MDs	**Total cost** (10^{-3})			
	Optimum	*DQN*	*PPO*	*Proposed method*
20	6.302	6.709	6.642	6.523
40	30.907	31.426	31.409	31.346
60	65.707	65.987	65.916	65.786
80	101.463	104.323	104.149	102.084
100	136.941	141.758	140.753	137.293

VI. CONCLUSION

In this article, we first design a blockchain-based framework to effectively secure the transactions between MDs and CSPs in an MCS system. Next, we propose a new DRL-based method to obtain the optimal policy for offloading computation-intensive tasks of PoW to edge in the blockchain-based MCS. Simulation results verify the effectiveness of the proposed method in achieving a low weighted cost of latency and power consumption. The proposed method outperforms benchmark methods and approximates the optimum with different numbers of MDs. In our future work, we intend to further improve the feasibility of the proposed blockchain-based MCS system by introducing the Delegated Proof-of-Stake (DPoS) consensus algorithm. Moreover, we plan to further consider the application of the blockchain to some important tasks such as the verification of trustworthy participants and sensory data to reduce the costs of using the blockchain. Besides, we will further consider the offloading costs in the design of the incentive mechanism.

ACKNOWLEDGMENTS

This work was partly supported by the National Natural Science Foundation of China under Grant No. 62202103, the Central Funds Guiding the Local Science and Technology Development under Grant No. 2022L3004, and the Funds for Scientific Research of Fujian Provincial Department of Finance under Grant No. 83021094.

REFERENCES

[1] D. Belli, S. Chessa, B. Kantarci, and L. Foschini, "Toward fog-based mobile crowdsensing systems: State of the art and opportunities," *IEEE ComMag*, vol. 57, no. 12, pp. 78–83, 2019.

[2] K. Antevski and C. J. Bernardos, "Federation in dynamic environments: Can blockchain be the solution?," *IEEE ComMag*, vol. 60, no. 2, pp. 32–38, 2022.

[3] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," *White Paper*, 2008.

[4] D. Borsatti, G. Davoli, W. Cerroni, and C. Raffaelli, "Enabling industrial iot as a service with multi-access edge computing," *IEEE ComMag*, vol. 59, no. 8, pp. 21–27, 2021.

[5] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE ComMag*, vol. 56, no. 8, pp. 14–19, 2018.

[6] H. Guo, J. Liu, and H. Qin, "Collaborative mobile edge computation offloading for iot over fiber-wireless networks," *IEEE Network*, vol. 32, no. 1, pp. 66–71, 2018.

[7] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actorcritic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[8] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things (IoT) Journal*, 2018.

[9] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2018.

[10] C. Li, J. Xia, F. Liu, D. Li, L. Fan, G. K. Karagiannidis, and A. Nallanathan, "Dynamic offloading for multiuser muti-cap mec networks: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology (TVT)*, vol. 70, no. 3, pp. 2922–2927, 2021.

[11] T. Maksymyuk, J. Gazda, M. Volosin, G. Bugar, D. Horvath, M. Klymash, and M. Dohler, "Blockchain-empowered framework for decentralized network management in 6g," *IEEE ComMag*, vol. 58, no. 9, pp. 86–92, 2020.

[12] Z. Ning, P. Dong, M. Wen, X. Wang, L. Guo, R. Y. Kwok, and H. V. Poor, "5g-enabled uav-to-community offloading: joint trajectory design and task scheduling," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 39, no. 11, pp. 3306–3320, 2021.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

BIOGRAPHIES

ZHEYI CHEN is currently a Professor with the College of Computer and Data Science at the Fuzhou University, China. He received his Ph.D. degree in Computer Science from the University of Exeter. His research interests include cloud-edge computing, resource optimization, deep learning, and reinforcement learning.

ZHENGXIN YU received her Ph.D. degree in Computer Science from the University of Exeter, UK. She is currently a senior research associate at the Lancaster University. Her research interests focus on federated deep learning, mobile edge computing, and cyber security.