Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
Cyber Technology—Article

Incentive-Aware Blockchain-Assisted Intelligent Edge Caching and Computation Offloading for IoT

Qian Wang [a,b], Siguang Chen [a,b,*], Meng Wu [a]

[a] *Jiangsu Key Laboratory of Broadband Wireless Communication and Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
[b] *School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*

ARTICLE INFO

Article history:
Received 28 March 2022
Revised 26 August 2022
Accepted 31 October 2022
Available online 4 January 2023

Keywords:
Computation offloading
Caching
Incentive
Blockchain
Federated deep reinforcement learning

ABSTRACT

The rapid development of artificial intelligence has pushed the Internet of Things (IoT) into a new stage. Facing with the explosive growth of data and the higher quality of service required by users, edge computing and caching are regarded as promising solutions. However, the resources in edge nodes (ENs) are not inexhaustible. In this paper, we propose an incentive-aware blockchain-assisted intelligent edge caching and computation offloading scheme for IoT, which is dedicated to providing a secure and intelligent solution for collaborative ENs in resource optimization and controls. Specifically, we jointly optimize offloading and caching decisions as well as computing and communication resources allocation to minimize the total cost for tasks completion in the EN. Furthermore, a blockchain incentive and contribution co-aware federated deep reinforcement learning algorithm is designed to solve this optimization problem. In this algorithm, we construct an incentive-aware blockchain-assisted collaboration mechanism which operates during local training, with the aim to strengthen the willingness of ENs to participate in collaboration with security guarantee. Meanwhile, a contribution-based federated aggregation method is developed, in which the aggregation weights of EN gradients are based on their contributions, thereby improving the training effect. Finally, compared with other baseline schemes, the numerical results prove that our scheme has an efficient optimization utility of resources with significant advantages in total cost reduction and caching performance.

© 2022 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

With the widespread popularity of the Internet of Things (IoT), our lives are undergoing a tremendous change, especially since the emergence of artificial intelligence (AI), which makes intelligence applications in an endless stream, such as smart transportation and smart healthcare. For instance, we can grasp all traffic conditions before trip and then choose a route with less traffic to the destination; we can also detect our physical states via smart wearable devices to achieve real-time monitoring of our health. These novel applications are being used by an increasing number of users and generate an enormous amount of data every day. According to Cisco statistics, approximately 2.5 exabytes (EB) of data is generated in one day [1]. Generally, the caching and computing resources of these devices are extremely limited, and they transfer the data (executing code, etc.) to the resource-rich but remote cloud center for completing tasks, but this method cannot meet the users' requirements for low delay and energy consumption. Edge computing effectively alleviates the above problems because the edge node (EN) is closer to users and equipped with much more resources than terminals.

Edge computing has become a promising computing paradigm in recent years, and how to efficiently utilize resources in the entire edge network is a hot topic [2]. For example, by optimizing offloading decisions, Wu et al. [3] minimized the cost of energy and delay, which sufficiently enhances the service experience of users. In addition to computing resource, the EN is equipped with certain caching resources. Zhang et al. [4] considered caching some contents required for task computing in the EN to reduce the transmission time, which minimizes the delay of all computation tasks by jointly optimizing offloading decision, caching decision, and resources allocation. In fact, there is usually more than one EN in the whole network, the resource usage and workload of each EN are different, these ENs can further improve resource utilization by collaborating

* Corresponding author.
E-mail address: sgchen@njupt.edu.cn (S. Chen).

https://doi.org/10.1016/j.eng.2022.10.014
2095-8099/© 2022 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

with each other. For example, Zhang et al. [5] proposed a new idea from the perspective of edge-to-edge collaborative offloading, which utilizes edge-to-edge and cloud-edge collaborative computation offloading to complete AI-based computation tasks and achieves delay minimization and efficient utilization of resources. However, in realistic scenarios, each EN is selfish, and ENs with sufficient resources usually do not voluntarily provide their own resources for free. Zhao et al. [6] and Zeng et al. [7] designed incentive-based collaboration mechanisms for vehicle tasks offloading, which gives rental fees to adjacent vehicles or edge servers to encourage them to assist the requester in completing tasks. Finally, the optimal task allocation strategy is obtained to minimize the system cost.

In the above studies, the solving methods they designed are based on the principles of traditional mathematical programming, Lyapunov or genetic algorithms, which are either not suitable for dynamic and complex network environments or the solving process relies on complex and costly conditions. Deep reinforcement learning (DRL) is very popular due to its flexible autonomic learning ability, and it breaks through the limitation of traditional mathematical programming algorithms for problem solving. However, centralized training requires a strong computing capability that usually occurs in the cloud center, and it will also lead to frequent data transmissions [8]. In addition, the node will inevitably be curious during collaboration, which brings about a risk of privacy leakage, especially for privacy-sensitive data such as diagnostic information of patients. As an emerging distributed learning method, federated learning (FL) provides a new perspective to overcome these problems. Instead of sharing the local original data, it only shares the gradient information to train the global model. Each agent (i.e., device or EN) utilizes local data for their own training, and then, they upload their gradients to the aggregation node, which is responsible for performing federated aggregation to update the global model. For example, Huang et al. [9] proposed a collaborative computation offloading scheme between small cell base stations based on federated DRL. By optimizing the offloading decision and interaction coordination, the minimum total energy consumption was obtained, which not only reduces the communication overhead in the training process but also protects the security of local training data in each small cell base station. In the research on edge computing for IoT, many good solutions for resource optimization are investigated, but the development of an intelligent, efficient, and secure resource optimization scheme still faces the following three challenges:

(1) Some of the current existing schemes have a simple optimization of resources and control factors. For example, they optimize offloading decision or caching decision with limited consideration of resources allocation; others have a more comprehensive consideration, but they are based on ideal scenarios. For example, the selected offloading nodes are always idle with no competitive relationship between them. Moreover, most of them assume that neighbor nodes voluntarily contribute their resources for free.

(2) Some schemes with collaboration considerations employ incentive mechanisms to strengthen the motivation of nodes to participate in collaboration. However, not all neighboring nodes are suitable as collaboration candidate nodes, that is, the collaborative nodes are not reasonably screened, thus reducing the quality of collaboration. In addition, the security of the collaboration is not guaranteed.

(3) Federated DRL-based algorithms in existing works are effective, but most of them employ simple average aggregation without accurately expressing the contribution of each agent to the system, and the training effect urgently needs to be improved.

Inspired by the above challenges, we propose a novel scheme, that is, an incentive-aware blockchain-assisted intelligent edge caching and computation offloading scheme for IoT, the specific contributions of which are as follows:

(1) With the aim of minimizing the total cost for completing all tasks in the EN, we jointly optimize the offloading decision, caching decision, computing resource allocation, and bandwidth allocation, in which the total cost includes delay, energy consumption, and collaboration costs. This approach achieves a comprehensive optimization of resources and control factors in the whole network and makes full use of collaboration between ENs to further improve resources utilization. Moreover, the pricing rule of collaboration cost based on task preference is more practical.

(2) Furthermore, we propose a blockchain incentive and contribution co-aware federated deep reinforcement learning (BICC-FDRL) algorithm, with an incentive-aware blockchain-assisted collaboration mechanism developed for local training. This mechanism can encourage all ENs to actively participate in collaboration with security guarantee. In this mechanism, we design a novel incentive method with low communication cost, and all qualified candidate ENs compete to become the final collaborator to obtain benefits.

(3) Particularly, to express the contribution of each agent in an accurate and direct way during the federated aggregation of BICC-FDRL, we present a contribution-based federated aggregation method. Through this method, the outstanding agents receive more attention. As a result, the training effect of the global model can be improved.

Finally, the experimental results verify that our proposed scheme has sufficient advantages in terms of caching performance, total cost reduction, and optimization utility.

The remainder of this article is organized as follows. Section 2 describes related works. The system model is constructed in Section 3, and the problem formulation is presented in Section 4. In Section 5, we design a BICC-FDRL algorithm. Subsequently, we present the performance evaluation in Section 6. Finally, conclusions are drawn in Section 7.

2. Related work

In recent years, edge computing for IoT has surged in popularity, and extensive works about its performance improvement via resource optimization have emerged. From the perspective of optimization variables (control and allocation of network resources), Refs. [10–15] minimized the cost of task completion. With joint optimization of offloading, computing, and communication resources, Chen et al. [10] minimized the energy consumption of completing tasks by using a dynamic voltage scaling technique and alternating minimization algorithm. Similarly, Malik and Vu [11] integrated wireless charging and computation offloading to minimize the energy consumption of the system. In view of the limited caching resource in EN, Liu et al. [12] developed a popular data caching scheme in edge computing that can find the approximate maximum caching revenue of a service provider by optimizing the caching decision. From a more comprehensive perspective, Refs. [13–15] took full advantage of computation offloading and caching to improve system performance. To minimize the average energy consumption of all users, Chen and Zhou [13] proposed a scheme with optimal offloading and caching decisions by employing a dynamic programming-based algorithm. Bi et al. [14] and Zhang et al. [15] jointly optimized offloading, service caching, and allocation of computing and communication resources. To minimize the weighted sum of delay and energy consumption, Bi et al. [14] solved the optimization problem in two steps: First, they derived a closed expression with optimal resource allocation and then optimized offloading and caching decisions with an alternate minimization algorithm. For multiusers with a multitask mobile edge computing system, Zhang et al. [15] investigated an algorithm based on semidefinite relaxation and alternating optimization to minimize system cost. The above studies improve the efficiency of task completion by utilizing the computing and caching capabilities of the EN, but the capability of a single EN is limited. In addition, if each EN only provides services for its local users, it will lead to an imbalanced workload and waste of resources.

To fill the above gap, Refs. [16–19] leveraged collaboration between ENs to further improve the utilization rate of resources and balance the workload. Ma et al. [16] and Zhong et al. [17] aimed to minimize the completion time of tasks. Ma et al. [16] investigated collaborative service caching and computation offloading among ENs, which optimizes the caching decision and offloading ratio of the task by a Gibbs sampling-based iteration algorithm. This scheme can adapt well to the heterogeneity of ENs. The collaboration scheme proposed in Ref. [17] is similar to that in Ref. [16], with additional consideration of resource allocation, and a modified generalized Benders decomposition algorithm was proposed to solve the optimization problem. To minimize the total delay and energy consumption of all users, Feng et al. [18] studied the collaborative data caching and offloading of neighboring ENs that can share caching data and computing resource, thus improving quality of service for users. Unlike the previous conventional collaboration caching between ENs, Yuan et al. [19] focused on the optimal number of collaborative ENs and forwarding groups, and an improved alternating direction multiplier method was proposed that can obtain the maximum cache hit rate with the constraint of low collaboration cost. These studies improve the utilization rate of network resources, but they assume that the service providers volunteer to provide computing or caching services to the requester. In fact, they are less motivated to participate in the collaboration process without an incentive.

In light of the shortcomings in the above studies, incentive mechanisms were designed in Refs. [20–24] to improve the collaborative motivation of users, ENs or clouds, which allows participants to benefit from it. An incentive mechanism between the cloud service operator and edge server was developed in Ref. [20]; furthermore, the authors designed a computation offloading scheme by jointly optimizing the offloading decision of the cloud operator and the payment of the edge server, so that it can maximize their utilities and achieve Nash equilibrium. Hou et al. [21] presented an edge-end incentive-driven task allocation scheme to maximize the system utility. In terms of the profiles and importance of the task in the device, it could be offloaded to its local edge server, a neighboring edge server, or a device cluster within the coverage of the same local edge server. In Ref. [22], the authors made full use of the computing resources of idle edge gateways and mobile devices, which formed edge clouds. Specifically, a resource trading model between edge clouds and mobile devices was built, which aims to maximize the profits of edge clouds and meet the computing requirements of mobile devices by leveraging market-based pricing and auction theory. Luo et al. [23] designed an efficient incentive mechanism in a device-to-device (D2D) network. According to the coalitional game, it established multiple micro-computing clusters among devices to achieve collaborative task computing, which enables each device to benefit from assisting in computing or relay and effectively reduces the global cost of the system. Additionally, for a D2D network, Zhang et al. [24] investigated a collaborative cache based on a multi-winner auction, which can obtain the maximal content caching revenue of all users with the optimal offloading and payment strategies and guarantee the profit fairness of auction winners. These schemes arouse network motility by designing an incentive mechanism (i.e., the nodes in the network can actively interact and make full use of the resources). However, the solving methods they proposed are all based on traditional iterative mathematical programming or game theory, the prerequisite of which is strict to obtain the optimal solution, making it unsuitable for dynamic and complex network environments. In addition, collaboration between nodes is neglected.

To construct a security- and intelligence-enabled network for adapting to dynamic and complex network, a machine learning-based approach was introduced. In Refs. [25–29], they combined the autonomous learning ability of DRL and the secure distributed computing of FL, which can overcome the drawbacks of the above schemes. For example, Refs. [25–27] proposed a double deep Q-network based FL algorithm. Zarandi and Tabassum [25] regarded a device as the agent and performed federated aggregation at the EN, it achieves task completion cost minimization on the terminal side by jointly optimizing offloading decision and the allocation of computing and communication resources. Study in Ref. [26] is similar to Ref. [25], and it had more detailed research and analysis on DRL and FL. Moreover, it gave additional consideration to caching optimization. Ren et al. [27] designed a reward and penalty mechanism between ENs and devices (i.e., devices will pay rewards to the EN that provides computing service, but the EN will be punished for task computing failure). This mechanism achieves the minimization of device payment by optimizing offloading decision and the allocation of energy units. Although FL can protect data privacy by keeping data locally, agents are vulnerable to malicious attacks in the process of uploading gradients. To further enhance data privacy protection, Cui et al. [28] introduced blockchain technology into FL and designed four smart contracts for each agent to ensure the security of their data. In addition, they compressed the gradient information uploaded by agents to reduce the communication overhead of aggregation and finally obtain excellent performance on the cache hit rate and security by optimizing the caching decision. The incentive idea also exists in blockchain technology, and Yu et al. [29] mentioned in their future work that the utilization of incentive idea in blockchain for resource borrowing not only strengthens the security of participant interactions but also plays a role in efficient resource utilization.

Based on the analysis of the aforementioned studies, we can see that they have their own advantages and provide us with great inspiration. However, these schemes still have room for improvement in the efficient utilization of resources and solution effects. It is not only necessary to comprehensively optimize the resources and control in the entire edge network but also very important to provide a secure and intelligent collaboration scheme with coexistence of competition and collaboration, which will bring about a significant improvement to system performance.

3. Network model

With the rapid development of IoT and AI techniques, various intelligent applications have been developed extensively, such as vehicle routing optimization in smart transportation and health monitoring in smart healthcare. These applications have led to the explosive growth of data generated by sensors, and users appear to have more stringent requirements for service delay and privacy protection [30]. Inspired by this, we design an incentive-aware blockchain-assisted intelligent edge computation offloading and caching model. As illustrated in Fig. 1, this model includes three layers: the user layer, edge layer, and cloud layer. The coverage area of the entire network is divided into M subareas, and each subarea includes N users and one EN, therefore, there are $M \times N$ users in the user layer and M ENs with certain capabilities of caching and computing in the edge layer.

When each user sends task requests to its local EN over the wireless link, the EN can either directly provide services to its local users or send requests for result sharing/computation offloading to

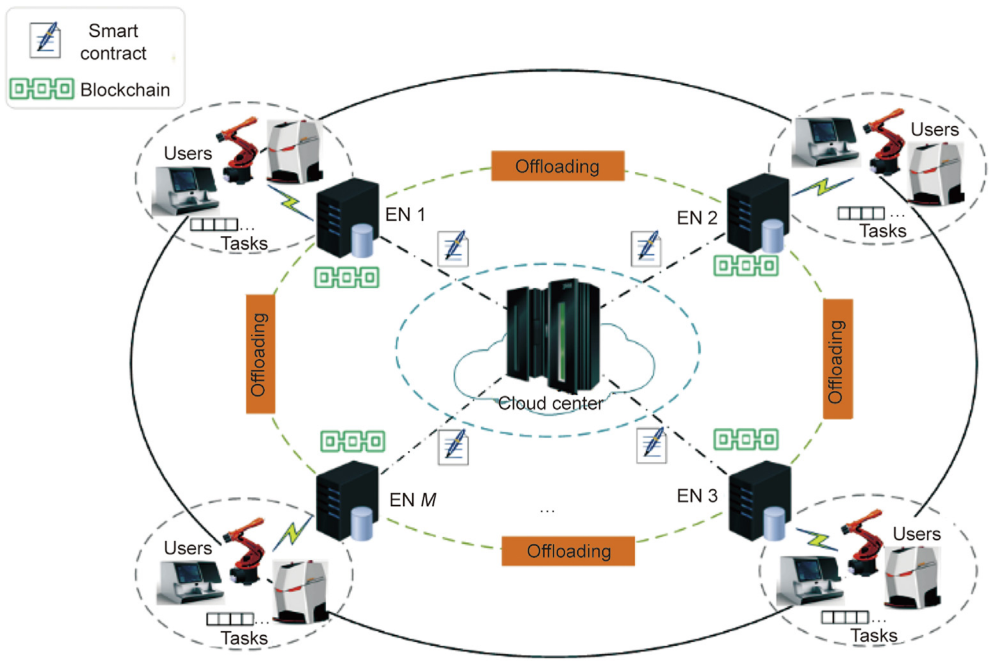

Fig. 1. Incentive-aware blockchain-assisted intelligent edge computational offloading and caching model. M: the number of subareas.

the neighboring EN, because neighbors can collaborate with each other. Moreover, to improve their motivation to participate in a beneficial and secure manner, incentive-aware blockchain-assisted collaboration is formed between ENs. In the process of their collaboration, the cloud center in the cloud layer will deploy the corresponding smart contract, which sets the trading rules for result sharing and computation offloading. The specific functions of each layer are defined as follows.

(1) **User layer**. Each user can generate one computation task at time slot t, and the same category task can be generated by multiple users. We denote that the task set generated by the whole user layer at a time slot t is $\Gamma = \{1, 2, \ldots, F\}$, that is, there are F categories of tasks. Due to the limited capabilities of users themselves, these tasks may be sent to their local ENs for efficient processing, so we define that the task set received by the EN m is $\Gamma_m, m \in \mathcal{M} = \{1, 2, \ldots, M\}$, $\mathcal{M}$ is the set of ENs.

(2) **Edge layer**. The edge layer is composed of M ENs, which are equipped with servers that have certain caching and computing capabilities. Here, ENs are regarded as agents, and they can use information collected from local users for model training. As the capability of a single EN is limited, neighbors can assist each other in improving the efficiencies of task processing and resource utilization, and it is beneficial for ENs that apply an incentive-aware blockchain-assisted collaboration model to this process. Specifically, when the EN receives the service request from the local user layer, if the result has been cached, then it will be returned directly to the user (i.e., a cache hit). Otherwise, the EN will send the request to the cloud center to seek result sharing from neighboring ENs, as a result, it will obtain an address of a tradable EN. According to the smart contract, it will pay the corresponding result sharing payment to the neighbor EN after receiving the result. If the result of the task is not cached anywhere, then the EN will decide whether to process it locally or offload to a neighboring EN. If the task needs to be offloaded, then it is similar to the process of cache sharing (i.e., sending a request to the cloud center for offloading task to a neighboring EN). Eventually, the EN will obtain the result and return it to the user. After each EN trains its model with local data, it uploads its model parameters to the cloud center for aggregation.

(3) **Cloud layer**. The cloud center of the cloud layer has two main functions: ① It assists collaborations between ENs and deploys smart contracts for them, which can be executed automatically, and ② it aggregates the model parameters of each EN. Specifically, when an EN sends a request of result sharing or computation offloading, the cloud center will broadcast its request to the entire blockchain network after its identity is verified to be legitimate and return the address of a tradable neighbor to the requesting EN. At the same time, the corresponding smart contract (trading rules) is deployed for the transaction between them. In addition, the cloud center is responsible for aggregating the training model parameters collected from ENs and then forwarding the aggregated parameters to each EN, which aims to update the local models in ENs.

3.1. Caching model

To return results to users as soon as possible and relieve the computing pressure of the EN, caching some results in the EN is a good approach. Furthermore, caching the computation result rather than the task itself not only saves the caching space but also improves the privacy protection of the users' original data. However, due to the limited caching space of the EN, it is impossible to cache all the computation results. Therefore, we consider results sharing between ENs to alleviate the above problem. Moreover, to improve the cache hit rate of results, they will be replaced regularly in ENs according to their popularities and preferences. The caching decision of task f is defined as α_m^f, if $\alpha_m^f = 1$, then it indicates caching the result of task f in EN m; otherwise, $\alpha_m^f = 0$. The detailed model of global popularity and EN preference for tasks are presented as follows.

In the beginning, the EN will cache computation results with high popularity in its cache pool in advance. The global popularity of tasks across the network follows the Mandelbrot Zipf distribution [31], which is defined as

$$P_f = \frac{(O_f + \tau)^{-\sigma}}{\sum_{i \in \Gamma} (O_i + \tau)^{-\sigma}}, \forall f \in \Gamma \tag{1}$$

where P_f is the global popularity of task f. O_f and O_i represent the ranking of the results for tasks f and i in descending order of global popularity, respectively. τ is denoted as the plateau factor, σ is the skewness factor, and i is the ith category task in Γ.

For EN m, we assume that the probability of receiving the service request of task f is δ_m^f ($\sum_{m\in\mathscr{M}}\delta_m^f=1$), and the preference of task f is $P_f^m=N_m^f/N$, which satisfies $\sum_{f\in\Gamma_m}P_f^m=1$ and $P_f^m=P_f\delta_m^f$, and N_m^f is the number of task f received by EN m. We denote the quantity set of all tasks received by EN m as $N_m=\left(N_m^f\right)^{1\times F}$.

When the result is not cached locally but in neighbors, with the aim of obtaining the result, the local EN pays the tradable neighbor EN n for the corresponding payment of result sharing, which we term the result sharing cost. The specific definition about the result sharing cost of task f (R_f^{cache}) is as follows:

$$R_f^{\text{cache}}=\lambda_f^{\text{cache}}\mathrm{e}^{-P_f^n}k_f,\ n\in\ \mathscr{M}\backslash\{m\} \tag{2}$$

where λ_f^{cache} represents the weight coefficient of the result sharing payment. Specifically, $\mathrm{e}^{-P_f^n}$ is viewed as a unit result sharing cost, which is inversely proportional to the preference of task f in EN n. k_f is the result size of task f.

3.2. Local model

When none of the ENs caches the result of task f, it is necessary to make an offloading decision $x_{m,f}^n$ for task f. If $\forall n\in\mathscr{M}\backslash\{m\}$, $x_{m,f}^n=0$, then task f will be computed at local EN m. We define the size of task f as s_f (bits), the central processing unit (CPU) cycles requested to complete the computation of task f as d_f (cycles), and the size of the computation result as k_f (bits). The completion time of task f at local EN m can be written as

$$t_f^m=\frac{d_f}{\beta_m^f u_m} \tag{3}$$

where u_m (cycles per second) is the computing capability of EN m, and β_m^f is denoted as the ratio of computing resource allocated to task f.

Consequently, the computing energy consumption $\left(e_f^m\right)$ of task f is as follows:

$$e_f^m=p_m^{\text{c}}t_f^m \tag{4}$$

where p_m^{c} is the computing power of EN m. Therefore, the cost of completing task f locally (R_f^{local}) can be represented as

$$R_f^{\text{local}}=\left(1-\sum_{n\in\mathscr{M}\backslash\{m\}}x_{m,f}^n\right)\left(\lambda_f^{\text{t}}t_f^m+\lambda_f^{\text{e}}e_f^m\right),n\in\mathscr{M}\backslash\{m\} \tag{5}$$

where λ_f^{t} and λ_f^{e} are the weight coefficients of time and energy costs, respectively.

3.3. Offloading model

When $x_{m,f}^n=1$, task f is offloaded from EN m to neighbor EN n. Particularly, there is no need for EN m to transmit task f to EN n when both EN m and n receive the service request of task f from their own local users at the same time slot t. In this way, EN m only needs to give a certain service payment R_f^{com} to the neighbor EN n for providing computing service, the payment of which is defined as

$$R_f^{\text{com}}=\lambda_f^{\text{com}}e^{-P_f^n}d_f \tag{6}$$

where λ_f^{com} represents the weight coefficient of the computation service sharing cost.

Otherwise, task f needs to be transferred to neighbor EN n, and its transmission time and the corresponding energy consumption are given as follows:

$$t_f^n=\frac{s_f}{v_{m,f}^n},f\notin\Gamma_m\cap\Gamma_n \tag{7}$$

$$e_f^n=p_m^{\text{t}}\frac{s_f}{v_{m,f}^n},f\notin\Gamma_m\cap\Gamma_n \tag{8}$$

where p_m^{t} is the transmission power of EN m, and $v_{m,f}^n$ represents the transmission rate of task f between EN m and n, which is defined as

$$v_{m,f}^n=\gamma_m^f B_m^n\log_2\left(1+\frac{p_m^{\text{t}}{h_m}^2}{w_m l_m^n\gamma_m^f B_m^n}\right) \tag{9}$$

where γ_m^f is the bandwidth ratio allocated to task f; B_m^n is the bandwidth size between EN m and n; h_m and w_m are the channel gain and noise of EN m, respectively; and l_m^n is the distance between m and n.

Therefore, the offloading cost of task f is represented as

$$R_{m,f}^n=\sum_{n\in\mathscr{M}\backslash\{m\}}x_{m,f}^n\times\begin{cases}\lambda_f^{\text{t}}t_f^m+\lambda_f^{\text{e}}e_f^m+R_f^{\text{com}},f\notin\Gamma_m\cap\Gamma_n\\ R_f^{\text{com}},f\in\Gamma_m\cap\Gamma_n\end{cases} \tag{10}$$

4. Problem formulation

According to the above model constructions and related definitions, in this incentive-aware blockchain-assisted edge computing network, as an important metric for system performance, the cost $C\left(\alpha_m^f,x_{m,f}^n,\beta_m^f,\gamma_m^f\right)$ of completing task $f\in\Gamma_m$ includes the costs of delay, energy consumption, and collaboration, which is defined as

$$C\left(\alpha_m^f,x_{m,f}^n,\beta_m^f,\gamma_m^f\right)=\left(1-a_m^f\right)\left(1-a_n^f\right)\left(R_f^{\text{local}}+R_{m,f}^n\right)+a_n^fR_f^{\text{cache}} \tag{11}$$

where a_m^f and a_n^f denote the caching states of task f's result in EN m and the neighbor EN n, respectively. a_m^f is represented as

$$a_m^f=\begin{cases}1,\alpha_m^f(t-1)=1\\ 0,\alpha_m^f(t-1)=0\end{cases} \tag{12}$$

where $\alpha_m^f(t-1)$ is the caching decision of task f at the previous time slot, and we define $a_m=\left(a_m^f\right)^{1\times F}$.

Consequently, the formulated optimization objective is shown in Eq. (13), which aims to minimize the total cost for completing all tasks received by EN m. By jointly optimizing caching decision α_m^f, offloading decision $x_{m,f}^n$, and the allocated ratios of computing and communication resources (i.e., β_m^f and γ_m^f), the optimization problem is formulated as follows:

$$\min_{\alpha_m^f,x_{m,f}^n,\beta_m^f,\gamma_m^f}\sum_{f\in\Gamma_m}C\left(\alpha_m^f,x_{m,f}^n,\beta_m^f,\gamma_m^f\right) \tag{13}$$

$$\text{s.t. } t_m^f\le t_m^{\max} \tag{13a}$$

$$e_m^f\le e_m^{\max} \tag{13b}$$

$$\sum_{f\in\Gamma_m}\alpha_m^f k_f\le C_m \tag{13c}$$

$$\sum_{f\in\Gamma_m}\left(1-x_{m,f}^{n}\right)\beta_m^f\le 1 \tag{13d}$$

$$\sum_{f\in\Gamma_m}x_{m,f}^{n}\gamma_m^f\le 1 \tag{13e}$$

$$\sum_{n\in\mathscr{M}}x_{m,f}^{n}\le 1 \tag{13f}$$

$$a_m^f x_{m,f}^{n}\neq 1 \tag{13g}$$

$$\alpha_m^f, x_{m,f}^{n}\in\{0,1\} \tag{13h}$$

where C_m represents the maximum caching size of EN m.

To ensure the quality and efficiency of service, the delay and energy consumption for completing task f should be constrained (i.e., constraints Eq. (13a) and Eq. (13b)). Because the limited resources (caching, computing, and bandwidth) of the EN are shared by multiple tasks, we give constraints Eqs. (13c)–(13e) that the total network resources allocated to all tasks do not exceed the maximum value. From a practical view, constraint Eq. (13f) demonstrates that task f can only be offloaded to at most one neighbor EN, and constraint Eq. (13g) means that the caching state and offloading decision of task f cannot be 1 simultaneously. The constraint Eq. (13h) indicates that caching decision α_m^f and offloading decision $x_{m,f}^{n}$ are both restricted to either 0 or 1.

Obviously, the above optimization problem is a mixed integer nonlinear optimization problem, and the variable dimension will be high when the network environment changes dynamically, and the number of users increases over time. As a result, it is difficult to solve directly by conventional methods, such as the gradient descent mathematical programming method and alternating direction method of multipliers. Recently, machine learning became a popular and vital artifice to solve this kind of resources allocation and decisions optimization problem in dynamic and time-varying networks, and it is viewed as an effective intelligent solving method. In view of the benefits of DRL and FL, which combines their characteristics of autonomic learning in complex and dynamic network environment, and the secure distributed computing, respectively, we will integrate these two machine learning ideas to solve the optimization problem Eq. (13).

5. BICC-FDRL

In this section, we propose a novel solving algorithm (i.e., BICC-FDRL algorithm, to achieve intelligent edge caching and computation offloading in the IoT). Inspired by the ideas of blockchain incentives, advantage actor–critic (A2C), and FL, the proposed algorithm mainly includes two parts: incentive-aware blockchain-assisted local training and contribution-based federated aggregation, the detailed framework of which is depicted in Fig. 2.

First, as an independent agent, each EN uses local information to train the local model by integrating the A2C concept. During the local training, an incentive-aware blockchain-assisted collaboration mechanism is developed to select appropriate collaborative neighbor EN, which will improve EN's motivation for participation in a beneficial and secure manner. We assume that the total rounds of local training in EN m is T_m, after EN m completes T_m training rounds, all ENs will send their local model parameters and corresponding average cumulative rewards to the cloud center simultaneously, which are utilized for aggregation operation and global model update. The cloud center calculates the contribution ratio of each EN as the aggregation weight of their model parameters and then aggregates the parameters to obtain new global model parameters. Finally, global model parameters are forwarded to all ENs to update their respective models. The detailed design and explanation of the above scheme are given in the following parts.

5.1. Incentive-aware blockchain-assisted local training

In general, we regard the joint optimization problem in the EN as a Markov decision process. It simulates an agent (i.e., EN) performing an action in the environment according to the current state so that it can change the environment state and obtain a reward. The process includes three key elements, which are defined as follows:

(1) **State.** The state space of EN m at time slot t is defined as $S_m(t)=\{N_m(t),a_m(t),\psi_m(t),\Im_m(t)\}$, where $N_m(t)$ represents the number of tasks received by EN m, and $a_m(t)$ is the cache state of EN m. $\psi_m(t)$ and $\Im_m(t)$ denote the remaining caching and computing resources in EN m, respectively, which are defined as

$$\psi_m(t)=C_m-\sum_{f\in\Gamma_m}\alpha_m^f k_f \tag{14}$$

$$\Im_m(t)=1-\sum_{f\in\Gamma_m}\left(1-x_{m,f}^{n}\right)\beta_m^f \tag{15}$$

(2) **Action.** We describe the action space of EN m at time slot t as $A_m(t)=\{\alpha_m(t),x_m(t),\beta_m(t),\gamma_m(t)\}$, which consists of caching decision set $\alpha_m(t)$, offloading decision set $x_m(t)$, and the allocated computing and bandwidth resource sets $\beta_m(t)$ and $\gamma_m(t)$. They are correspondingly defined as

$$\alpha_m(t)=\{\alpha_m^1(t),\alpha_m^2(t),\dots,\alpha_m^F(t)\} \tag{16}$$

$$x_m(t)=\left\{x_{m,1}^{m}(t),x_{m,2}^{m}(t),\dots,x_{m,F}^{m}(t)\right\} \tag{17}$$

$$\beta_m(t)=\{\beta_m^1(t),\beta_m^2(t),\dots,\beta_m^F(t)\} \tag{18}$$

$$\gamma_m(t)=\{\gamma_m^1(t),\gamma_m^2(t),\dots,\gamma_m^F(t)\} \tag{19}$$

(3) **Reward.** The learning objective of EN is to maximize the cumulative reward, while the objective of our optimization problem is to minimize the total cost. Therefore, we set the immediate reward as a negative exponential of the total cost:

$$r_m(t)=\begin{cases}\mathrm{e}^{-\sum_{f\in\Gamma_m}\mathrm{cost}_f}, \text{if Eq.(13a)} - \text{Eq.(13h)}\\ \varepsilon, \text{else}\end{cases} \tag{20}$$

where $\mathrm{cost}_f=C\left(\alpha_m^f,x_{m,f}^{n},\beta_m^f,\gamma_m^f\right)$, and ε ($\varepsilon<0$) is a penalty value given by the specific environment. Therefore, the cumulative reward is $R_m=\sum_{t=1}^{T_m}r_m(t)$.

In our scheme, each EN acts as a DRL agent and trains its model based on local information. Inspired by the A2C concept, the designed network architecture of the local model in Fig. 2 consists of two subnetworks (i.e., the actor network and the critic network). The actor network will select action according to the current state and then interact with the environment. Then, the environment will give reward $r_m(t)$ and transfer to the next state. Particularly, reward $r_m(t)$ is obtained according to action $A_m(t)$ and A2C blockchain-assisted collaboration mechanism. The critic network is responsible for the action evaluation of actor network based on the current and next states. Unlike the traditional A2C design, our critic network uses the advantage function to evaluate actor performance instead of a simple value function. Because the evaluation of the action is not just based on how well it is but also depends on the environment improvement with the action.

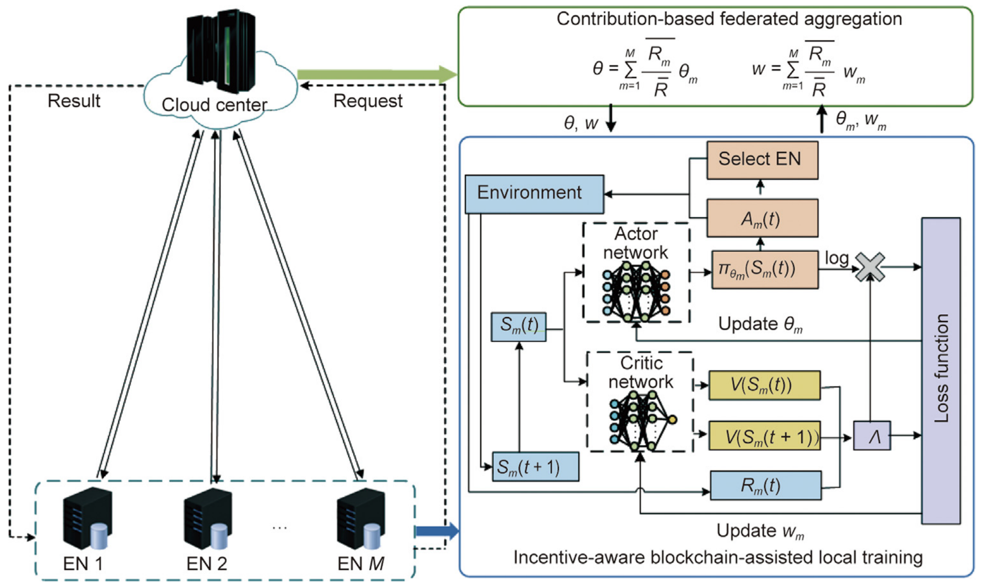

Fig. 2. The framework of the BICC-FDRL. $\overline{R_m}$: the average cumulative reward of completing tasks in EN m; $\bar{R}$: the sum of the average cumulative rewards of all edge nodes; θ_m: the parameter of actor network in EN m; w_m: the parameter of critic network in EN m; θ: the parameter of actor network in the global model; w: the parameter of critic network in the global model; $S_m(t)$: the state space of EN m at time slot t; $S_m(t+1)$: the state space of EN m at time slot $t+1$; $A_m(t)$: action space of EN m at time slot t; $\pi_{\theta_m}(S_m(t))$: the policy generated by actor network; $R_m(t)$: the cumulative reward of EN m at time slot t; $V(S_m(t))$: the value function of the current state; $V(S_m(t+1))$: the value function of the next state; Λ: the advantage function.

At time slot t, the actor network generates policy $\pi_{\theta_m}(S_m(t))$ based on the current state $S_m(t)$ of local EN m. Because the outputs of the actor network are continuous values, the decisions of caching and offloading in $A_m(t)$ are discrete variables. Therefore, we need to approximate the policy $\pi_{\theta_m}(S_m(t))$ so that the caching and offloading decisions in it are converted into discrete values, which are defined as

$$\alpha_m(t) = x_m(t) = \begin{cases} 0, \alpha_m(t), x_m(t) < 0.5 \\ 1, \alpha_m(t), x_m(t) \geq 0.5 \end{cases} \tag{21}$$

With the aim of inspiring each EN to participate in result sharing and computation offloading processes in a beneficial and secure manner, we develop an incentive-aware blockchain-assisted collaboration mechanism as described in Fig. 3. This mechanism will select one neighbor EN that can provide service to local EN, and the value of $x^n_{m,f}$ will be determined.

According to current state $S_m(t)$ and the action $A_m(t)$, when the result is not cached in local EN ($a^f_m = 0$) or local computing cannot meet user requirements ($x^m_{m,f} = 0$) , the tradable neighbor EN will be selected for obtaining the task result, and then local EN will give corresponding payment to the neighbor who provides the service. Therefore, the reward $r_m(t)$ is calculated based on $A_m(t)$ and cost of this collaboration mechanism. Different from other blockchain-based incentive methods, our method is suitable for the scenario in this paper. The implementation of this method is closely related to edge caching and computing offloading, and they promote each other. There are two collaboration cases between ENs, which are explained as follows:

(1) **Result sharing.** By encouraging each EN to cache its preferred task results, this incentive-aware blockchain-assisted collaboration can not only meet the local demand but also help other neighbors, thus improving cache utilization.

When the result is not cached in the local EN, the local EN will send a request of result sharing to the cloud center. The cloud center will broadcast its request across the whole blockchain network after verifying that its identity is legitimate so that it can know which neighboring ENs cache this result, that is, $a^f_n(t) = 1, n \in \mathscr{M}\backslash\{m\}$. These neighboring nodes will become candidates, which are sorted according to the preference of task f from low to high.

Next, the cloud center deploys the smart contract for the transaction. That is, to save as much costs as possible, neighbor n with the lowest result sharing cost R^{cache}_f will be chosen. According to Eq. (2), R^{cache}_f is a negative exponential function of P^n_f. In other words, EN m will choose the neighboring EN n with a higher preference for task f. Now, the cost$_f$ in $r_m(t)$ is defined as cost$_f = R^{\text{cache}}_f$.

Subsequently, ENs complete the legitimate transaction of sharing results based on smart contracts. Then, the neighbor EN n records the transaction process as a block. Finally, it will broadcast this block to all candidates to reach a consensus. Compared with broadcasting to all ENs, this method will reduce the communication overhead.

(2) **Computation offloading.** By encouraging the local EN to select the tradable neighboring EN with a similar preference or more sufficient computing resources for computation offloading, this incentive-aware blockchain-assisted collaboration will improve the task completion efficiency.

Similar to the process of result sharing, when task f in the local EN m is offloaded to the neighboring EN for computing (i.e., $\sum\limits_{n\in\mathscr{M}\backslash\{m\}} x^n_{m,f} = 1$), the neighboring EN providing computation service will be given a certain payment, but candidates are chosen in different ways. The neighboring ENs are arranged in descending order according to the amount of available computing resources, and the neighbors that exceed the average computing resource are regarded as candidates. Then, the neighbor EN n with the lowest offloading cost $R^n_{m,f}$ is selected. From Eq. (10), we know that the neighboring EN that receives the same request and has a higher preference of task f is more likely to be selected. Now, the cost$_f$ in $r_m(t)$ is defined as cost$_f = R^n_{m,f}$.

After the above process, reward $r_m(t)$ is given by the environment, and the environment state transfers to the next state $S_m(t+1)$. Next, we input the current state and the next state into

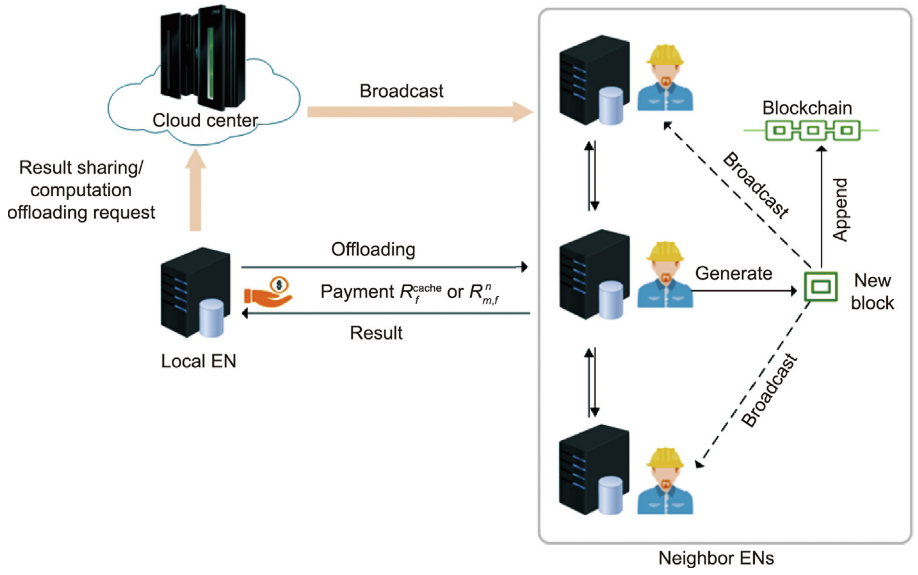

Fig. 3. Incentive-aware blockchain-assisted collaboration mechanism.

the critic network, and their value functions $V(S_m(t))$ and $V(S_m(t+1))$ are the outputs. Based on these, we can calculate advantage function Λ as

$$\Lambda = Q(t) - V(S_m(t)) \tag{22}$$

$$Q(t) \approx R_m(t) + \rho V(S_m(t+1)) \tag{23}$$

where $Q(t)$ represents the expected value function after performing action $A_m(t)$ at the current state $S_m(t)$, and ρ is a discount factor. If $\Lambda > 0$, then it means that performing action $A_m(t)$ will bring a positive incentive to the system state. Moreover, the larger the Λ value is, the greater the improvement in the network. In addition, the utilization of the advantage function can reduce the variance in evaluation and accelerate the convergence rate.

During the training, we use the Adam optimizer to update the model parameters. The gradient of the policy loss function in the actor network is expressed as $\Lambda \nabla_{\theta_m} \log \pi_{\theta_m}(S_m(t))$, and the gradient of the value loss function in the critic network is expressed as $\Lambda \nabla_{w_m} V(S_m(t)$. Finally, the update processes of these two networks are described as

$$\theta_m \rightarrow \theta_m + \eta \Lambda \nabla_{\theta_m} \log \pi_{\theta_m}(S_m(t)) \tag{24}$$

$$w_m \rightarrow w_m + \mu \Lambda \nabla_{w_m} V(S_m(t)) \tag{25}$$

where η and μ are the learning rates of the actor network and critic network, respectively.

5.2. Contribution-based federated aggregation

An important step in FL is the aggregation of model parameters. That is, when each EN has been trained for several rounds, the ENs that participated in FL will upload their trained model parameters to the cloud center for aggregation simultaneously.

The most classic aggregation method is federated averaging (FedAvg) aggregation [32], proposed by Google, which adopts a simple average aggregation based on the training sample size of the EN. The model parameter aggregation operation of FedAvg is shown as follows:

$$\theta = \sum_{m=1}^{M} \frac{K_m}{M} \theta_m \tag{26}$$

$$w = \sum_{m=1}^{M} \frac{K_m}{M} w_m \tag{27}$$

where θ and w are parameters of the actor network and critic network in the global model, respectively. We assume that there are M ENs participating in FL and the sample size of agent m is K_m.

Generally, sample size is used as the basis for aggregation because more training samples will introduce a better model training effect. However, it is not inevitable. To depict the model training effect of ENs more accurately, we consider utilizing a more direct method (i.e., cumulative reward R_m). A higher cumulative reward indicates that the network model parameters of EN are better, and a greater contribution will be made to the performance improvement of the future global model.

Motivated by this, we design a contribution-based federated aggregation method, which refers to the average cumulative reward ratio $\overline{R_m}/\overline{R}$ of EN. During federated aggregation, each EN needs to upload their contribution and model parameters to the cloud center, and the cloud center aggregates these parameters to obtain the parameters of the global model, which are defined as

$$\theta = \sum_{m=1}^{M} \frac{\overline{R_m}}{\overline{R}} \theta_m \tag{28}$$

$$w = \sum_{m=1}^{M} \frac{\overline{R_m}}{\overline{R}} w_m \tag{29}$$

where $\overline{R_m} = R_m/|\Gamma_m|$ is the average cumulative reward of completing tasks, and $\overline{R} = \sum_{m=1}^{M} \overline{R_m}$. Moreover, the definition of contribution avoids the unfair evaluation caused by the different number of tasks in different ENs.

After the process of aggregation, the cloud center will forward the aggregation parameters of the global model to each EN for updating their local model. Each EN keeps learning until the model converges to the optimal strategy.

To provide a clearer understanding of the solution processing for the objective, we summarize it as Algorithm 1.

Algorithm 1. BICC-FDRL algorithm for edge caching and computation offloading.

```
Input: the total rounds of local training $T_m$
    the total episodes of global training $T$
    the global popularity of tasks $P_f$
    the probability $\delta_m^f$ of service request for task $f$
Output: Minimum total cost with optimal $A_m{}^*$
Initialization: $\theta_m = \theta$, $w_m = w$
For episode $\ell = 1$ to $T$ do
  For each EN $m \in \mathscr{M}$ do
    For $t = 1$ to $T_m$ do
      Based on $S_m(t)$, perform $A_m(t)$
      Select collaborate neighbor EN $n$ based on $A_m(t)$ and
      cost$_f$ of incentive-aware blockchain-assisted
      collaboration mechanism
      Obtain $r_m(t)$ and transfer to next state $S_m(t+1)$
      Perform $S_m(t) = S_m(t+1)$
      Calculate $\Lambda$, the gradient of policy loss function
      $\Lambda \nabla_{\theta_m} \log \pi_{\theta_m}(S_m(t))$ and the gradient of value loss
      function $\Lambda \nabla_{w_m} V(S_m(t)$
      Update $\theta_m$ and $w_m$ with Eqs. (24) and (25), respectively
    End for
  End for
  Perform contribution-based federated aggregation based on
  Eqs. (28) and (29)
  Forward global model parameters to each EN, that is,
  $\theta_m = \theta$, $w_m = w$
End for
Obtain the minimum total cost with optimal $A_m{}^*$
```

6. Performance evaluation

In this section, we will evaluate the performance of the proposed scheme by conducting simulation experiments, which mainly consist of two parts: convergence analysis of the BICC-FDRL algorithm in Section 6.1 and advantage analysis of the proposed scheme in Section 6.2.

To construct simulations, we set $M = 7$ ENs and $N = 25$ users within the communication coverage of each EN. For convenience, each EN has the same computing capability and caching size (i.e., 2.5×10^9 cycles·s^{-1} and 6 Mbit, respectively), and the bandwidth between ENs is uniformly set to 100 MHz. There are $F = 50$ tasks generated by the whole user layer at time slot t, the range of the task size is $[30, 50]$ Mbit, and the range of its result size is $[300, 500]$ Kbit. Moreover, the plateau factor and the skewness factor of task popularity are set to $\tau = -0.95$ and $\sigma = 0.50$, respectively. Particularly, we configure that the ENs have different preferences and demands, this differentiated consideration can reflect the heterogeneity of ENs.

For each EN, its actor network and critic network are both three-layer fully connected neural networks, and the hidden layer with 64 units have a rectified linear unit (ReLU) activation function. In the output layer of the actor network, the offloading and caching decisions use softmax as their activation function, while others use a sigmoid activation function. In addition, we add a two-layer fully connected neural network as the shared input layer for the actor and critic networks, which is used to abstract complex system states. The computational complexity of DRL mainly depends on the structure and the parameter number of the neural network, which is measured by the requirement of floating-point operations (FLOPs). In this paper, all the neural networks in our algorithm requires 0.20 million FLOPs (MFLOPs) to process a state input.

6.1. Convergence analysis of BICC-FDRL algorithm

In Fig. 4, we study the convergence of cumulative reward for different learning rates of the actor network. It can be seen from the figure that the convergence speed of the curve with $\eta = 10^{-5}$ is obviously lower than that of the other two curves with a higher learning rate. Although the cumulative reward with $\eta = 10^{-4}$ converges slower than that when $\eta = 10^{-3}$, the former is slightly more stable, and their final convergence values are extremely close. Actually, after many experiments, we find that the rapid learning rate will become trapped in the local optimal solution and the slow learning rate will not converge to the optimal solution in a finite time. Therefore, in the following experiments, we set the learning rate of the actor network to 10^{-4}.

The influence of different network learning rates on the convergence of loss is shown in Fig. 5. The loss function value can reflect the training effect and stability of the model. As the model is trained increasingly better, the loss decreases and eventually stabilizes to a small value (the ideal loss is 0). When $\mu = 0.050$ and $\mu = 0.010$, their loss convergence performance is obviously better than that when $\mu = 0.001$. Similar to the above analysis, a higher learning rate has better convergence performance in a certain range. Finally, we set $\mu = 0.050$ after many experiments.

The convergence of loss for different aggregation methods is depicted in Fig. 6. From the curves of the two aggregation methods, the loss of our proposed contribution-based federated aggregation is lower than that of FedAvg, which proves that our aggregation method has a better model training effect. Because in our aggregation method, the EN with a good learning effect has great reference value, it contributes more to the system and therefore has a greater aggregation weight, which improves the global training effect in the future and makes the policy more accurate.

The training effect with different numbers of aggregated ENs is depicted in Fig. 7, that is, the number of ENs participating in federated aggregation is 3, 5, and 7. The model with seven participating ENs has the highest cumulative reward. Furthermore, the oscillation amplitude of the three curves proves that more ENs participating in FL will make the model training more stable. As the number of ENs increases, the global model learns more information in the whole network (i.e., when more training samples are available, the training works better). However, if too many ENs participate in federated aggregation and upload their model parameters to the cloud center, then a communication burden will occur. Through more experiments, we find that the model convergence performance does not change significantly when the number of

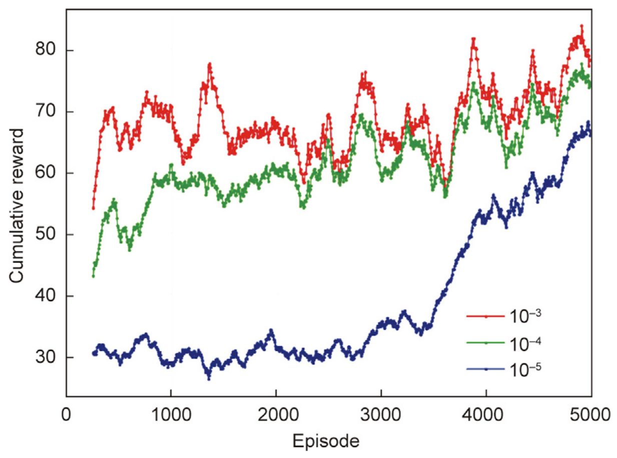

Fig. 4. The reward convergence of the actor network with different learning rates.

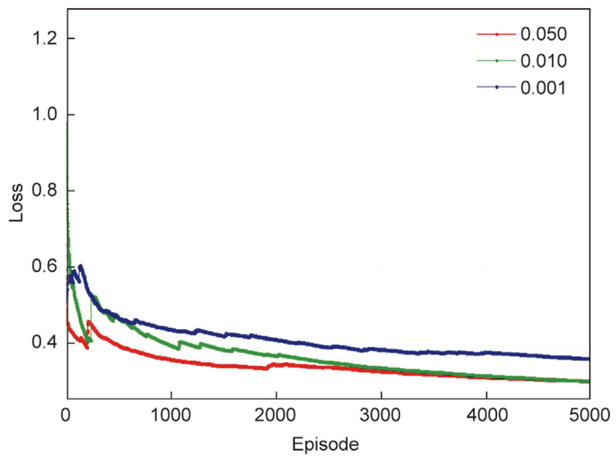

Fig. 5. The loss convergence of the critic network with different learning rates.

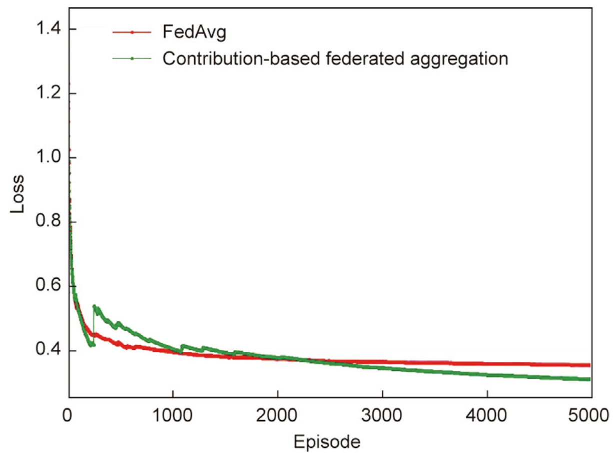

Fig. 6. The loss convergence of different aggregation methods.

ENs reaches a certain value. In our network scenario, we set $M = 7$ ENs.

6.2. Advantage analysis of the proposed scheme

We will analyze the cache performance advantage of our proposed scheme by comparing it with the other three caching schemes, that is, least recently used (LRU), LRU-2, and least frequently used (LFU).

As the hit rate and the total cost are the key evaluation metrics to show the performance of caching schemes, the following simulations will analyze the cache performance of the four schemes from these two aspects. The definition of hit rate H is as follows:

$$H = \frac{\sum_{f \in \Gamma_m} \varphi_f}{|\Gamma_m|} \tag{30}$$

where

$$\varphi_f = \begin{cases} 0, \sum_{m \in \mathcal{M}} a_m^f = 0 \\ 1, \text{else} \end{cases} \tag{31}$$

Fig. 8 shows the hit rate of different caching schemes with varying EN cache sizes. As the cache size increases, EN can cache more results, so the hit rate of the four schemes increases gradually. The hit rates ranked from high to low are our proposed scheme, LRU-2, LFU, and LRU. The hit rate of LRU is the lowest, which is mainly due to the popular results being replaced by the accidental unpopular results. The third-ranked LFU caches the result with high request frequency, which indicates a high preference, but it takes a long time to learn well. The second-ranked LRU-2 combines the advantages of LRU and first in first out, but it still cannot cache popular results accurately. Our proposed scheme has the highest hit rate because of its ability to cache results based on result popularity and the preferences of different EN regions. Moreover, the incentive-aware blockchain-assisted collaboration mechanism is designed in our scheme, which motivates the willingness of ENs to share results and then improves the hit rate.

Fig. 9 describes the total cost of different caching schemes with the cache size of EN varying from 1 to 6 Mbit. Corresponding to Fig. 8, the larger the cache size of the EN is, the higher the hit rate, which saves the cost of completing the task. Therefore, the total cost of the four schemes shows a decreasing trend, and our proposed scheme is the lowest compared with others. In addition, when the cache size reaches a certain value, the hit rate reaches convergence, so the total cost will be stabilized.

According to the above simulations, we know that our proposed scheme has a better caching performance, which brings about a significant advantage to the total cost reduction.

To analyze the advantage of our proposed scheme from a broader perspective, we compare it with three baseline schemes

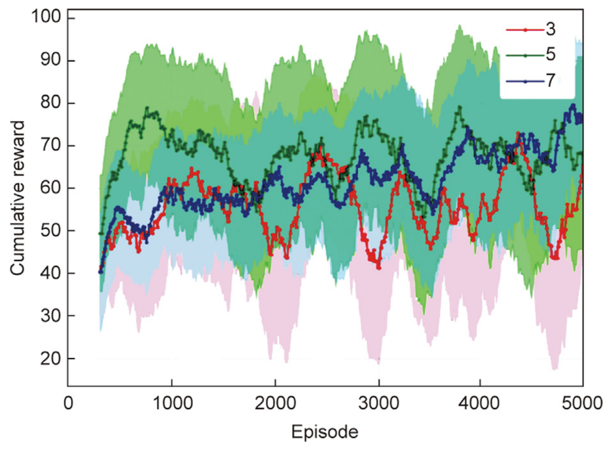

Fig. 7. The training effect with different numbers of aggregated ENs.

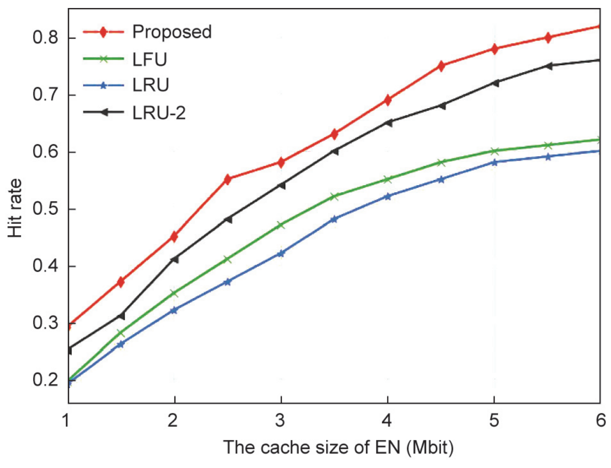

Fig. 8. Hit rate of different caching schemes.

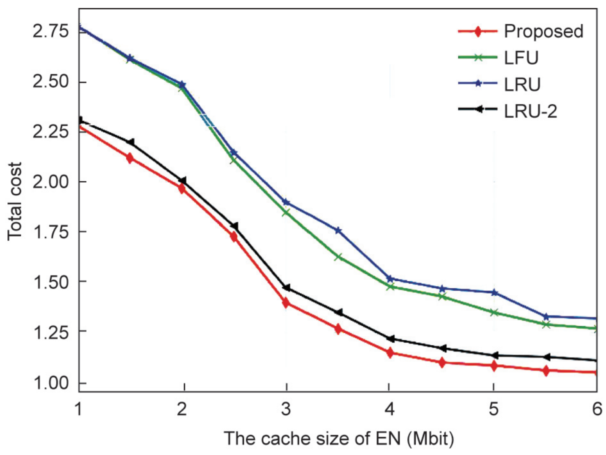

Fig. 9. Total cost of different caching schemes.

(i.e., federated deep reinforcement learning-based cooperative edge caching (FADE) [33], efficient and flexible management: a federated deep Q network approach (EFMFDQN) [34], and centralized A2C). In FADE, it adopts a popularity-based caching method similar to ours, but does not consider offloading decisions and other resource optimizations. EFMFDQN jointly optimizes the offloading decisions, bandwidth allocation ratio and transmit power by a deep *Q*-network based FL algorithm. It should be noted that centralized A2C is the optimal baseline scheme with minimum total cost under the ideal condition. Specifically, it regards the cloud center as an agent, which collects the information of the whole network to train the model centrally, and deploys the A2C algorithm to optimize the total reward.

For the above four schemes, their total costs under different numbers of tasks are compared in Fig. 10. The performance of our scheme is close to that of centralized A2C and outperforms the other two schemes. Specifically, when there are few tasks, the caching resources of all ENs are sufficient to cache the results of these tasks without local or offloading computation. Therefore, the total cost of the other three schemes is lower than EFMFDQN, which does not consider cache optimization. With the increasing number of tasks, the advantages of our scheme become increasingly obvious. Since the caching space of EN is finite, the total cost of FADE is the highest, which only considers the caching decision optimization. This finding occurs because our proposed scheme gives a more comprehensive consideration of optimization factors (i.e., joint optimization of offloading decision, caching decision, computing and communication resources allocation). Moreover, we design an incentive-aware blockchain-assisted mechanism that promotes collaboration between ENs and reduces collaboration costs.

From a more intuitive perspective, we will use the optimization utility to analyze the advantage of our proposed scheme, and the optimization utilities of the four schemes are defined as follows:

$$U_{\text{BICC-FDRL}} = \frac{C_{\text{local}} - C^*_{\text{BICC-FDRL}}}{C_{\text{local}}} \tag{32}$$

$$U_{\text{FADE}} = \frac{C_{\text{local}} - C^*_{\text{FADE}}}{C_{\text{local}}} \tag{33}$$

$$U_{\text{EFMFDQN}} = \frac{C_{\text{local}} - C^*_{\text{EFMFDQN}}}{C_{\text{local}}} \tag{34}$$

$$U_{\text{centralized A2C}} = \frac{C_{\text{local}} - C^*_{\text{centralized A2C}}}{C_{\text{local}}} \tag{35}$$

where $C^*_{\text{BICC-FDRL}}$, C^*_{FADE}, C^*_{EFMFDQN}, and $C^*_{\text{centralized A2C}}$ represent the optimal total cost obtained by BICC-FDRL, FADE, EFMFDQN, and centralized A2C, respectively; and C_{local} is the total cost of local computing. $U_{\text{BICC-FDRL}}$, U_{FADE}, U_{EFMFDQN}, and $U_{\text{centralized A2C}}$ are the optimization utilities of the BICC-FDRL, FADE, EFMFDQN, and centralized A2C, respectively.

From Fig. 11, we can observe that when the number of ENs increases, the optimization utility will increase. Obviously, the number of ENs increases implies that there are more caching and computing resources in the network, which enhances collaboration between ENs and then improves resource utilization. Moreover, as the number of ENs increases, the number of ENs participating in federated aggregation also increases, similar to Fig. 7, the training effect will be improved. When there are nine ENs in the network, the utility of our scheme is almost equal to centralized A2C and significantly better than that of the other two schemes. This simulation reveals that our scheme efficiently utilizes caching, communication and computing resources.

It is worth noting that, despite the centralize A2C has optimal result in Figs. 10 and 11, our scheme is still meaningful. Specifically, in our scheme, A2C algorithm is combined with FL, it can sink the training to the ENs, which alleviates the computing pressure of the cloud center and communication pressure. Meanwhile, owing to the training mode of FL, the model update only transmits the model parameters instead of the training data, which has the effect of privacy protection.

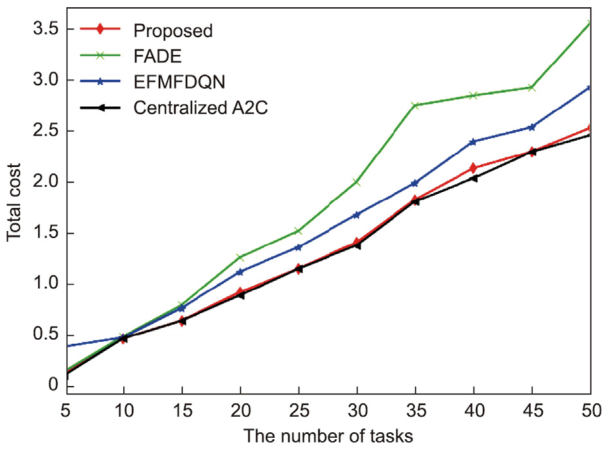

Fig. 10. Total cost of different schemes under different numbers of tasks.

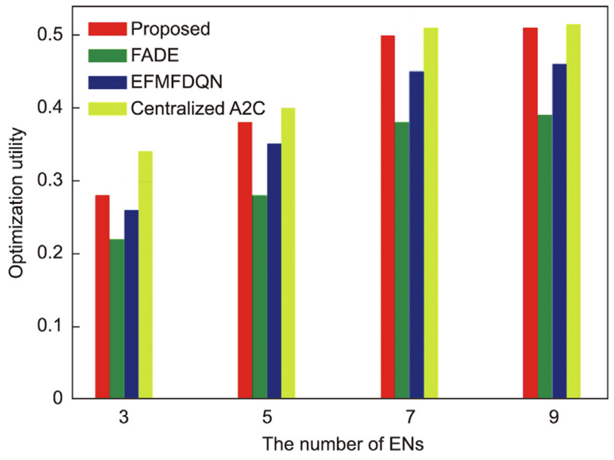

Fig. 11. The optimization utility of different schemes under different sizes of the network.

7. Conclusions

In this paper, we propose an incentive-aware blockchain-assisted intelligent edge caching and computation offloading scheme for IoT. To minimize the total cost of completing tasks in EN, we provide a comprehensive optimization of offloading, caching, and resources allocation. Particularly, task preference-based pricing rules are conducive for cost saving. Furthermore, to obtain the optimal solution, we develop a BICC-FDRL algorithm. In this algorithm, incentive-aware blockchain-assisted local training provides a secure incentive mechanism for encouraging ENs to collaborate, that is, incentive-aware blockchain-assisted collaboration mechanism, all ENs participate in result sharing and computation offloading actively for benefit. Moreover, for improving the training effect, we design a contribution-based federated aggregation method by paying more attention to ENs with large contributions. The numerical results indicate that our proposed scheme has significant performance advantages in terms of hit rate, total cost and optimization utility compared with other baseline schemes, such as FADE and EFMFDQN. The BICC-FDRL algorithm developed in our work is suitable for the scenario that all ENs have homogeneous learning models. However, this FL method has limitations when ENs have heterogeneous learning models. In future work, we will investigate a more efficient personalized FL with heterogeneous models.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (61971235), the China Postdoctoral Science Foundation (2018M630590), the Jiangsu Planned Projects for Postdoctoral Research Funds (2021K501C), the 333 High-level Talents Training Project of Jiangsu Province, the 1311 Talents Plan of Nanjing University of Posts and Telecommunications, and the Jiangsu Planned for Postgraduate Research Innovation (KYCX22_1017).

Compliance with ethics guidelines

Qian Wang, Siguang Chen, and Meng Wu declare that they have no conflict of interest or financial conflicts to disclose.

References

[1] Evans D. The Internet of Things: how the next evolution of the Internet is changing everything. Report. San Jose: CISCO; 2011.

[2] Chen S, Zhu X, Zhang H, Zhao C, Yang G, Wang K. Efficient privacy preserving data collection and computation offloading for fog-assisted IoT. IEEE Trans Sustain Comput 2020;5(4):526–40.

[3] Wu F, Liu X, Li H, Fan Q, Zhu L, Wang X, et al. Energy–time efficient task offloading for mobile edge computing in hot-spot scenarios. In: Proceedings of the IEEE International Conference on Communications; 2021 Jun 14–23; Montreal, QC, Canada; 2021.

[4] Zhang J, Hu X, Ning Z, Ngai ECH, Zhou L, Wei J, et al. Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching. IEEE Internet Things J 2019;6(3):4283–94.

[5] Zhang L, Wu J, Mumtaz S, Li J, Gacanin H, Rodrigues JJPC. Edge-to-edge cooperative artificial intelligence in smart cities with on-demand learning offloading. In: Proceedings of the IEEE Global Communications Conference (GLOBECOM); 2019 Dec 9–13; Waikoloa, HI, USA; 2019.

[6] Zhao L, Yang K, Tan Z, Song H, Al-Dubai A, Zomaya AY, et al. Vehicular computation offloading for industrial mobile edge computing. IEEE Trans Ind Inform 2021;17(11):7871–81.

[7] Zeng F, Chen Q, Meng L, Wu J. Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing. IEEE Trans Intell Transp Syst 2021;22(6):3247–357.

[8] Zhao Z, Feng C, Yang HH, Luo X. Federated-learning-enabled intelligent fog radio access networks: fundamental theory, key techniques, and future trends. IEEE Wirel Commun 2020;27(2):22–8.

[9] Huang X, Leng S, Maharjan S, Yan Z. Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks. IEEE Trans Veh Technol 2021;70(9):9282–93.

[10] Chen S, Zheng Y, Lu W, Varadarajan V, Wang K. Energy-optimal dynamic computation offloading for industrial IoT in fog computing. IEEE Trans Green Commun Netw 2020;4(2):566–76.

[11] Malik R, Vu M. On-request wireless charging and partial computation offloading in multi-access edge computing systems. IEEE Trans Wirel Commun 2021;20(10):6665–79.

[12] Liu Y, He Q, Zheng D, Zhang M, Chen F, Zhang B. Data caching optimization in the edge computing environment. In: Proceedings of the IEEE International Conference on Web Services (ICWS); 2019 Jul 8–13; Milan, Italy; 2019. p. 99–106.

[13] Chen Z, Zhou Z. Dynamic task caching and computation offloading for mobile edge computing. In: Proceedings of the IEEE Global Communications Conference; 2020 Dec 7–11; Taipei, China; 2020.

[14] Bi S, Huang L, Zhang YJA. Joint optimization of service caching placement and computation offloading in mobile edge computing systems. IEEE Trans Wirel Commun 2020;19(7):4947–63.

[15] Zhang G, Zhang S, Zhang W, Shen Z, Wang L. Joint service caching, computation offloading and resource allocation in mobile edge computing systems. IEEE Trans Wirel Commun 2021;20(8):5288–300.

[16] Ma X, Zhou A, Zhang S, Wang S. Cooperative service caching and workload scheduling in mobile edge computing. In: Proceedings of the IEEE Conference on Computer Communications; 2020 Jul 6–9; Toronto, ON, Canada; 2020. p. 2076–85.

[17] Zhong S, Guo S, Yu H, Wang Q. Cooperative service caching and computation offloading in multi-access edge computing. Comput Netw 2021;189:107916.

[18] Feng H, Guo S, Yang L, Yang Y. Collaborative data caching and computation offloading for multi-service mobile edge computing. IEEE Trans Veh Technol 2021;70(9):9408–22.

[19] Yuan P, Shao S, Geng L, Zhao X. Caching hit ratio maximization in mobile edge computing with node cooperation. Comput Netw 2021;200:108507.

[20] Liu Y, Xu C, Zhan Y, Liu Z, Guan J, Zhang H. Incentive mechanism for computation offloading using edge computing: a stackelberg game approach. Comput Netw 2017;129:399–409.

[21] Hou W, Wen H, Zhang N, Wu J, Lei W, Zhao R. Incentive-driven task allocation for collaborative edge computing in industrial Internet of Things. IEEE Internet Things J 2022;9(1):706–18.

[22] Wang Q, Guo S, Wang Y, Yang Y. Incentive mechanism for edge cloud profit maximization in mobile edge computing. In: Proceedings of the IEEE International Conference on Communications (ICC); 2019 May 20–24; Shanghai, China; 2019.

[23] Luo S, Chen X, Zhou Z, Chen X, Wu W. Incentive-aware micro computing cluster formation for cooperative fog computing. IEEE Trans Wirel Commun 2020;19(4):2643–57.

[24] Zhang T, Fang X, Liu Y, Li GY, Xu W. D2D-enabled mobile user edge caching: a multi-winner auction approach. IEEE Trans Veh Technol 2019;68(12):12314–28.

[25] Zarandi S, Tabassum H. Federated double deep Q-learning for joint delay and energy minimization in IoT networks. In: Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops); 2021 Jun 14–23; Montreal, QC, Canada; 2021.

[26] Wang X, Han Y, Wang C, Zhao Q, Chen X, Chen M. In-edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. IEEE Netw 2019;33(5):156–65.

[27] Ren J, Wang H, Hou T, Zheng S, Tang C. Federated learning-based computation offloading optimization in edge computing-supported Internet of Things. IEEE Access 2019;7:69194–201.

[28] Cui L, Su X, Ming Z, Chen Z, Yang S, Zhou Y, et al. CREAT: blockchain-assisted compression algorithm of federated learning for content caching in edge computing. IEEE Internet Things J 2022;9(16):14151–61.

[29] Yu S, Chen X, Zhou Z, Gong X, Wu D. When deep reinforcement learning meets federated learning: intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. IEEE Internet Things J 2021;8(4):2238–51.

[30] Chen S, Yang L, Zhao C, Varadarajan V, Wang K. Double-blockchain assisted secure and anonymous data aggregation for fog-enabled smart grid. Engineering 2022;8:159–69.

[31] Hefeeda M, Saleh O. Traffic modeling and proportional partial caching for peer-to-peer systems. IEEE/ACM Trans Netw 2008;16(6):1447–60.

[32] McMahan HB, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the International Conference on Artificial Intelligence and Statistics; 2017 Apr 20–22; Fort Lauderdale, FL, USA; 2017. p. 1273–82.

[33] Wang X, Wang C, Li X, Leung VCM, Taleb T. Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching. IEEE Internet Things J 2020;7(10):9441–55.

[34] Guo YH, Zhao ZC, He K, Lai SW, Xia JJ, Fan LS. Efficient and flexible management for Industrial Internet of Things: a federated learning approach. Comput Netw 2021;192(4):108122.