Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 7212984, 15 pages
https://doi.org/10.1155/2022/7212984

Research Article

Collaborative Caching in Edge Computing via Federated Learning and Deep Reinforcement Learning

Yali Wang [1,2] and Jiachao Chen [1]

[1]*School of Computer and Information Engineering, Henan Normal University, Xinxiang, Henan, China*
[2]*Engineering Lab of Intelligence Business and Internet of Things, Henan, Xinxiang 453007, China*

Correspondence should be addressed to Yali Wang; 121071@htu.edu.cn

Received 11 October 2022; Revised 29 November 2022; Accepted 6 December 2022; Published 22 December 2022

Academic Editor: Dharmendra Singh Rajput

By deploying resources in the vicinity of users, edge caching can substantially reduce the latency for users to retrieve content and relieve the pressure on the backbone network. Due to the capacity limitation of caching and the dynamic nature of user requests, how to allocate caching resources reasonably must be considered. Some edge caching studies improve network performance by predicting content popularity and actively caching the most popular content, thereby ignoring the privacy and security issues caused by the need to collect user information at the central unit. To this end, a collaborative caching strategy based on federated learning is proposed. First, federated learning is used to make distributed predictions of the preferences of users in the nodes to develop an effective content caching policy. Then, the problem of allocating caching resources to optimize the cost of video providers is formulated as a Markov decision process, and a reinforcement learning method is used to optimize the caching decisions. Compared with several basic caching strategies in terms of cache hit rate, transmission delay, and cost, the simulation results show that the proposed content caching strategy reduces the cost of video providers, and has higher cache hit rate and lower average transmission delay.

1. Introduction

The explosive growth of mobile devices, including cell phones, wearable devices, connected cars, and Internet-of-Things (IoT) devices, has led to exponential growth in data traffic. As a result, great pressure has been exerted on the backhaul network, with resultant increases in network latency. Resource-constrained mobile devices also face considerable challenges in supporting computation-intensive and time-critical applications, such as video services, voice control, gesture recognition, 3D modeling, natural language processing, and online interactive games. On the other hand, video data is also growing explosively. With substantial storage space and powerful computing ability, cloud data centers are the best content repository for some video vendors, such as YouTube and TikTok. However, mobile devices suffer performance degradation when retrieving all data from the cloud.

To address these issues, edge computing (EC), which is a promising paradigm, has been introduced to provide a service environment with computing and caching capacity. By hiring computing and storage resources on edge servers, mobile apps and content vendors (referred to as vendors hereafter) can host their apps and content on edge servers to provide low-latency and high-quality services for users [1]. In this way, EC can greatly alleviate the congestion of the backhaul network, improve the quality of user experience (QoE) by meeting the strict requirement of response delay, and enhance location awareness.

Compared to cloud computing, MEC is still constrained by limited storage capacity [2]. Edge servers can cache only part of the content, and video users can retrieve data from nearby edge servers instead of from remote cloud servers if the data are already cached on those edge servers [3]. Edge servers are prone to storing popular content to obtain higher hit ratios. Thus, content popularity prediction [4] and cache cooperation strategies between edge servers are important to improve caching performance.

In recent years, traditional caching strategies, such as the least recent use (LRU) and least frequent use (LFU) approaches, have been extensively studied [5, 6]. However, these methods are not very efficient because they do not take content popularity into account. Most of the existing work on edge caching handles the popularity based on the assumption that the content popularity is known in advance, which is impractical. Therefore, it is imperative to improve the caching efficiency by properly predicting content popularity [7]. Recently, centralized machine learning can learn on its own from available data and make accurate decisions or predictions on unseen data with the help of algorithms [8, 9]. In literature [10], an optimal cache resource allocation scheme during the next update period is drawn using a neural network from the content history requests collected from all users. In literature [11], a deep learning-based content popularity prediction scheme is proposed. In literature [12], the authors propose a popularity prediction model for each content category in terms of historical popularity by training a simplified bidirectional long and short-term memory (Bi-LSTM) network. In literature [13], the authors propose an evolutionary learning-based content caching strategy that adaptively learns content popularity over time. In literature [14], a user preference model is proposed to predict content popularity and track popularity changes based on user preferences and the features of the requested content.

However, most existing caching schemes require users to share their private preference data with a central server, which may pose privacy and security risks [15]. Federated learning (FL) is a distributed framework that learns global models on a server while protecting participants' privacy by allowing each participant to share locally trained model parameters with the server instead of their local data. A mechanism to protect user privacy is introduced in the literature [16], where the content popularity obtained by the user's local model is weighted with user preferences. The authors in the literature [17] propose an FL-based model aggregation method that divides all users into multiple subspaces based on their contextual information.

In addition, most of the above articles consider the user's perspective to optimize user request latency and improve the QoE. However, for content providers, how to increase profit and reduce costs are additional factors to consider. In literature [18], the multicell collaborative caching problem is studied to minimize the total cost for content providers. The literature [19] considers the caching resource allocation problem in a scenario where a network operator coexists with multiple content providers. This problem is modeled as a multileader multifollower Stackelberg game, and the optimal caching strategy for the content providers is obtained.

In this paper, we propose a cooperative caching strategy in edge caching by considering the cost of optimizing content providers and the privacy security of users. Aiming to ensure the privacy security of users, we adopt FL to predict user preferences in different edge nodes in a distributed manner and apply the approach to the design of the caching policy. For the cost problem for content providers, we use a deep reinforcement learning (DRL) based collaborative caching (Dueling DDQN) algorithm, which combines DDQN and Dueling DQN to make caching decisions for edge nodes. The contributions of this paper are as follows:

(i) Due to the dynamic changes in video popularity on the network, the FL framework is used to accurately predict the content popularity in a given region. The proposed model can learn the influence of multidimensional content features to formulate an effective caching strategy

(ii) To reduce the cost of video providers, a video collaborative caching strategy based on deep reinforcement learning is proposed. We construct a collaborative edge cache model based on transforming storage and energy consumption into costs. By combining DDQN and Dueling DQN, the proposed algorithm can effectively reduce the high valuation of DQN and accelerate the convergence speed. The experimental results verifies the validity of our approach

(iii) Experimental verification and comparative analysis are conducted on the proposed cache strategy, and our strategy is compared with other baseline algorithms. The experimental results show that our strategy optimizes the cost of video providers and has a high cache hit ratio

The rest of this paper is organized as follows: Section 2 summarizes the relevant work; Section 3 defines the system model and problem description; Section 4 details the proposed cache collaboration scheme; Section 5 uses simulation results to evaluate the cache performance; finally, Section 6 summarizes this paper.

2. Related Work

Many different strategies and algorithms have been for edge caching. In [20, 21], the user can only obtain the requested content from the local base station or cloud data center, while the noncooperative content caching scheme that cannot be obtained from the nonlocal base station is proposed. Since the cache capacity of a single base station is limited, to alleviate the storage capacity limitation of edge cache nodes, a collaborative cache strategy is adopted to solve the problem that limited cache node capacity affects cache efficiency and degrades system performance. Literature [22] proposes a cooperative cache strategy for heterogeneous cellular networks by transforming the optimal strategy design of content caching into an integer linear programming problem that is solved by the subgradient method. Literature [23] describes the problem of optimal collaborative content caching as a 0-1 integer program to minimize the average download delay. A greedy algorithm based on content popularity is used to solve the problem. Compared with the popular cache strategy, this strategy can substantially improve the content cache hit ratio and reduce the average content delivery delay.

Due to the dynamic nature of the network and the complexity of the environment, an efficient cache strategy must take user requirements into account. However, accurate user requirements are difficult to obtain. Literature [10–14] design cache strategies based on machine learning technology, estimating the future needs of users according to the content popularity and user preference data set with dynamic characteristics of the time series. Literature [24] proposes an online active cache scheme using the bidirectional deep recursive neural network (BRNN) model to predict time series content requests and update the edge cache accordingly. Literature [25] proposes a collaborative cache strategy of edge servers based on the software-defined network (SDN), in which multilayer sensory neural networks are used to predict video content request probability by mobile users and to construct an objective minimization function to maximize the utilization of edge servers' resources. The approach adopts a branch-and-bound algorithm to determine the optimal global solution. Literature [26] proposes a deep reinforcement learning approach and an improved branch delimitation strategy to solve the problem of jointly optimizing cooperative edge caching and wireless resource allocation in IoT networks, respectively. However, centralized access to user data may lead to user privacy exposure. Federated learning is a distributed machine learning framework that can effectively solve this problem [16, 17]. Thus, it is necessary to extract, analyze, and identify content popularity, user preferences, and user movement patterns while respecting user privacy. Literature [27] proposes a federated K-means scheme for privacy protection that is used for active caching in the next-generation cellular network. This scheme protects user privacy by means of two privacy-protection technologies: FL and secret sharing. Literature [28, 29] propose an intelligent F-RANS framework based on FL that can accurately predict the content popularity distribution in a network by applying FL to user demand prediction.

Due to the limited storage capacity of edge servers, it is impossible to guarantee that all content provided by the content provider (CP) will be cached to the edge server, and a user's request will directly affect the income of the CP. Therefore, the study of collaborative cache strategy must also consider maximizing the income of the CP [18, 19]. To reduce the delay and cost in the cloud-side collaboration environment, literature [30] proposes a content collaboration caching strategy. The strategy considers the delay gain and cache cost gain brought by the cached content to the cooperative domain and designs the cached content according to the gain. Because of the considerable request delay problems and high operation costs in the current video service-based caching strategy, literature [31] proposes a moving edge computing video caching strategy with coordinated optimization of delay and energy consumption. The method adopts a branch-and-bound algorithm to solve the optimization problem with the aim of reducing request delays for users and lowering costs for suppliers. This paper proposes an edge cache strategy algorithm based on reinforcement learning that considers cache placement and privacy security while protecting user privacy and reducing the costs of video providers. The recent work summary can be seen in Table 1.

3. System Architecture

In this section, we introduce a collaborative edge caching system architecture that supports FL frameworks. The architecture consists of a network model, content request model, local popularity model, cooperative cache model, and request cost model. The main parameter symbols are listed in Table 2.

3.1. Network Model.

Figure 1 shows the system scenario. This scenario contains a set of BSs $M\{1, 2, \cdots, f, \cdots, M\}$, and UEs $U\{1, 2, \cdots, u, \cdots, U\}$. Each BS can serve multiple UEs that have disjoint coverage areas. Mobile edge servers are deployed at each BS to provide edge caching service for UEs and to make wireless resource allocation decisions. Let s_M be the caching capacity of BS$_m$; then, all BSs capacity sets can be defined as $S = \{s_1, s_2, \cdots, s_M\}$. All the MEC servers can exchange cache information and share data through backhaul links and the cache manager (CM). Mobile devices send content requests at the beginning of each time slot t.

3.2. Content Request Model.

The content repository set located in the cloud server is denoted as $F = \{1, 2, \cdots, F\}$, and the size of content $f \in F$ is denoted as z_f. The set of requesting users at BS_m in time slot t is defined as U_m^t, and the number of UEs requesting content from BS_m during time slot t is defined as N_m^t. Each UE is assumed to be associated with only one BS in each time slot, i.e., each UE can be served by only one BS in a given time slot. The number of requests from all users can be defined as $R_{u,m}^t = [R_{1,m}^t, R_{2,m}^t, \cdots, R_{u,m}^t, \cdots, R_{U,m}^t]$.

3.3. Local Popularity Model.

Due to the diversity of the content preferences in different BSs, the local content popularity of all content in BS$_m$ at time slot t is defined as a content popularity vector $P_m^t = [P_{m,1}^t, P_{m,1}^t, \cdots, P_{m,f}^t, \cdots, P_{m,F}^t]$. In addition, considering the privacy and security of users, FL is applied to accurately predict content popularity without the UE uploading all individual user preference data to the BS.

3.4. Cooperative Cache Model.

The content requests of mobile devices are first received by BSs. If the requested content has been cached in the local BS, it will be pushed to users immediately. A binary local content delivery variable $x_{u,m,f}^t \in \{0, 1\}$ indicates whether the local BS provides services for the UE: $x_{u,m,f}^t = 1$ if a response is requested and $x_{u,m,f}^t = 0$ otherwise. If the requested content is not cached in the local BS, the BS will obtain the requested content from other BSs through the CM. Let $x_{u,i,f}^t \in \{0, 1\} (i \in M, i \neq m)$ denote a nonlocal BS serving the UE: $x_{u,i,f}^t = 1$ if another BS responds to the request and $x_{u,i,f}^t = 0$ otherwise. If the CM cannot find the requested content in any BS, the local BS will obtain the requested content from the cloud server

Table 1: Comparison of existing papers addressing edge caching problems.

Reference	Optimization objective	Method	Disadvantages
[22]	Download latency	Hungarian algorithm	No quantitative benefits
[23]	Cache hit ratio	Greedy algorithm	High complexity
[24]	Cache hit ratio	Bidirectional recurrent neural networks	Privacy security
[25]	Energy consumption	Branch and bound algorithm	Privacy security
[26]	Estimating content popularity	Federated k-means scheme	High complexity
[27]	Minimize traffic cost	Federated learning	Lower model accuracy
[29]	User response latency	Heuristic algorithm	Homogeneous user demand distribution
[30]	The cost of the video provider	Branch and bound algorithm	High time consuming

Table 2: Key notations and descriptions.

Notation	Description
M	Set of BSs
U	Set of UE
F	Set of content
U_m^t	Set of UE requesting content from BS_m in period t
$R_{u,m}^t$	Request vector for each UE in slot t
P_m^t	Local content popularity for all content in BS_m in time slot t
$x_{u,m,f}^t, x_{u,i,f}^t, x_{u,c,f}^t$	Content delivery variables via local, collaborative and cloud
$d_{m,f}$	Caching decisions for content f
z_f	Size of content
p	Energy consumption of each bit of data cached by MEC server
$p_{u,m}, g_{u,m}$	The transmission power from BS_m to UE_u, the channel gain between BS_m and UE_u
σ^2	The variance of additive Gaussian white noise
$\phi_{o,m}, \phi_{c,o}$	Transmission rate from BS_m to UE_u, transfer rate from CM to cloud server
$E_{m,f}, E_{i,f}, E_{c,f}$	Costs delivered via on-premises, collaboration and cloud
$\cos \mathrm{t}_{\mathrm{total}}, \cos \mathrm{t}_{ca}, \cos \mathrm{t}_{\mathrm{tran}}$	Total cost, cache cost, transmission cost
x_f^n	The feature vector of the content f
y_f	The category label of the content f
$p_{t,u,f}, \widehat{p}_{t,u,f}$	The probability that user u in time slot t requests content f, the predicted probability that user u in time slot t requests content f
R_u	The cumulative number of requested samples for user u
$w_u, w_u^{r+1} w_m$	The user preference model parameter vector of user u, the user preference model parameter vector learned by user u at the r-th iteration. The parameter vector of the regional integrated model
$\mathrm{Loss}\left(w_u, x_f, y_f\right)$	The logistic loss of user u
$g_r g_{1:r}$	The gradient vector of the r-th sample with respect to w_u's logistic loss, the sum of the gradient vectors of the logistic loss of the first r samples
$\lambda_1, \lambda_2; \alpha, \beta$	Positive regularization parameter; tuning parameters
$\eta_r, \eta_{r,n}, \sigma$	The nonincreasing learning rate, the learning rate of the n-th feature, and the parameter related to the learning rate η_r
N	Feature dimension

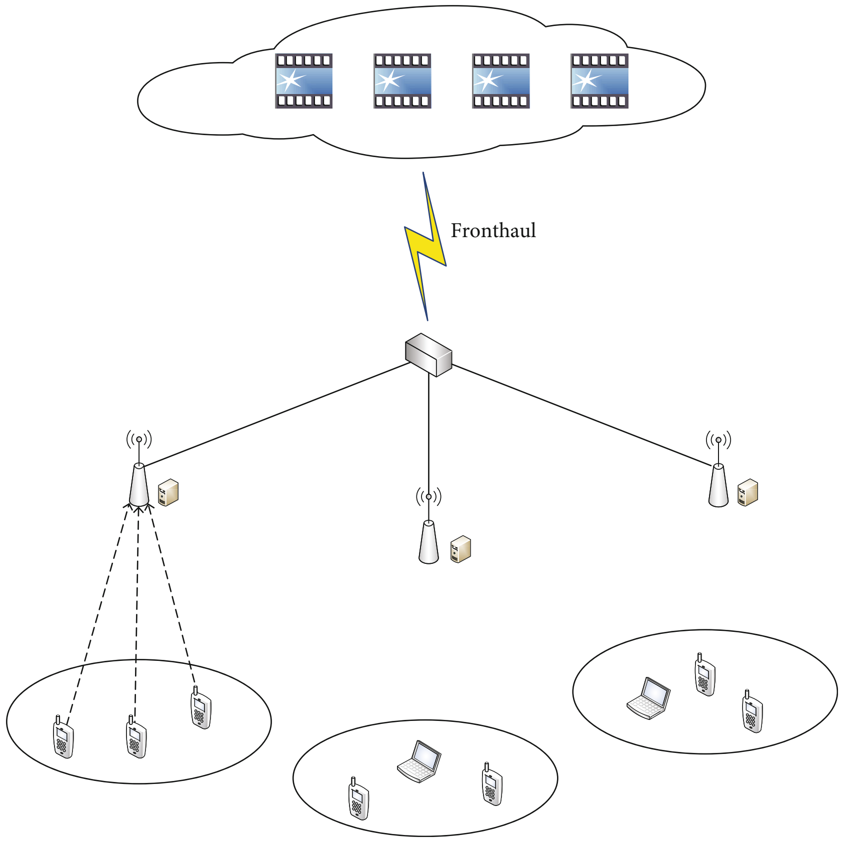


Figure 1: Illustration of the edge cache scenarios.

and deliver it to the UE. Let $x^t_{u,c,f} \in \{0, 1\}$ denote whether the UE obtains the content from cloud server c in time slot t: $x^t_{u,i,f} = 1$ if the UE obtains content f from the cloud server and $x^t_{u,i,f} = 0$ otherwise.

3.5. Request Cost Model. The cost of the system is composed of two main parts: the storage energy consumption and the transmission energy consumption of the content on the MEC. If the content is cached in the BS, there will be additional storage energy consumption. Assuming that all BS servers have the same performance, the storage cost of BS caching content [31] within period T can be expressed as

$$\cos \mathrm{t}_{ca} = \sum_{f=1}^{F} d_{m,f} \cdot z_f \cdot p \cdot T \tag{1}$$

where the caching decision $d_{m,f} \in \{0, 1\}$ indicates whether content f is cached in BS_m and p is the energy consumption of the MEC server to cache each bit of data. The wireless transmission rate between BS_m and UE_u can be obtained by the Shannon formula:

$$\phi_{u,m} = B \log_2 \left(1 + \frac{p_{u,m} g_{u,m}}{\sigma^2} \right), \tag{2}$$

where B represents the bandwidth of the base station, $p_{u,m}$ denotes the transmission power from BS_m to UE_u, $g_{u,m}$ is the channel gain between BS_m and UE_u, and σ^2 is the variance of additive Gaussian white noise. Then, the transmission energy consumption of UE_u to download content f from its local BS_m is

$$E_{m,f} = \frac{z_f}{\phi_{u,m}} \cdot p_e, \tag{3}$$

where $z_f / \phi_{u,m}$ denotes the time taken for file f to be transmitted from the user to the local base station, p_e is the transmission power between BSs. We define the transmission rate between a BS and the CM as $\phi_{o,m}$ and that between the CM and the cloud server as $\phi_{c,o}$. The transmission energy consumption of UE acquiring content from nonlocal BSs is expressed as:

$$E_{i,f} = \frac{2z_f}{\phi_{c,m}} \cdot p_e + E_{m,f}, \tag{4}$$

where $z_f / \phi_{c,m}$ denotes the time taken for file f to be transmitted from CM to the nonlocal base station. The energy consumption of UE downloading content f from cloud server c can be expressed as:

$$E_{c,f} = \frac{z_f}{\phi_{c,o}} \cdot p_c + \frac{z_f}{\phi_{o,m}} \cdot p_e + E_{m,f}, \tag{5}$$

where p_c is the transmission power of the cloud server. $\phi_{o,m} > \phi_{c,o}$ can be seen from the transmission rate, so the transmission energy consumption is $E_{c,f} > E_{i,f} > E_{m,f}$. Therefore, the transmission cost can be expressed as follows:

$$\cos \mathrm{t}_{\mathrm{tran}} = \sum_{u=1}^{N_m^t} \sum_{f=1}^{F} \left(x_{u,m,f}^t \cdot E_{m,f} + x_{u,i,f}^t \cdot E_{i,f} + x_{u,c,f}^t \cdot E_{c,f} \right). \tag{6}$$

Thus, the total cost can be expressed as

$$\cos \mathrm{t}_{\mathrm{total}} = \cos \mathrm{t}_{ca} + \cos \mathrm{t}_{\mathrm{tran}}. \tag{7}$$

3.6. Formalization. In this paper, the provider cost-based edge cocaching approach aims to minimize the total cost by efficiently caching content on edge servers. This problem can be expressed mathematically as:

$$\begin{aligned} \min \quad & \sum_{m=1}^{M} \cos \mathrm{t}_{\mathrm{total}}, \\ s.t. \quad & \mathrm{C1} : d_{m,f} \in \{0,1\}, \\ & \mathrm{C2} : x_{u,m,f}^t, x_{u,i,f}^t, x_{u,c,f}^t \in \{0,1\}, \\ & \mathrm{C3} : \sum_{f=1}^{F} d_{m,f} \cdot z_f \le s_m. \end{aligned} \tag{8}$$

where constraints C1 and C2 indicate that the cache decision and content delivery variables are binary. C3 indicates that the data in each BS should not exceed its storage capacity.

4. Problem Solution

In this section, we predict the user's request behavior via the factor machine (FM) algorithm [32] to account for different users' personality preferences in different scenarios. The FM algorithm can solve the feature combination problem under sparse data conditions. In addition, considering users' privacy security, we accurately predict the content popularity by applying FL to the content popularity prediction algorithm, which does not require the UE to upload all individual user preference data to the BS.

4.1. FL-Based Content Popularity Prediction Model

4.1.1. Creation of the Local Model. For each content f, define $x_f^n = \{x_f^1, x_f^2, \cdots, x_f^d\}$ as its feature vector. y_f is the category tag. If the content is requested, $y_f = 1$; otherwise, $y_f = 0$. We define $p_{t,u,f}$ as the probability of requesting content f for UE$_u$ in time slot t. The correspondence between the feature vector for the requested content and the category label is approximated based on the sigmoid function, and the FM model represents the user preferences. The formula is expressed as follows:

$$\hat{p}_{t,u,f} = \frac{1}{1 - e^{\hat{y}_f}}, \tag{9}$$

$$\hat{y}_f = w_{u,0} + \sum_{i=1}^{d} w_{u,i} x_{f,i} + \sum_{i=1}^{d} \sum_{j=i+1}^{d} w_{u,ij} x_{f,i} x_{f,j}, \tag{10}$$

where d denotes the sample feature dimension and $x_{f,i}$ is the value of the i-th feature of content f. $w_{u,0} w_{u,i} w_{u,ij}$ are model parameters. In the case of sparse data, very few samples will satisfy the nonzero cross term. When the number of training samples is insufficient, insufficient and inaccurate training of the parameters $w_{u,ij}$ is likely, which affects the model's effectiveness. Therefore, an e-dimensional auxiliary vector $v_i = (v_{i,1}, v_{i,2} \cdots v_{i,l} \cdots v_{i,e}), l \in [1, e]$ is introduced for each feature $x_{f,i}$. Then, the second-order parameter can be expressed as $w_{u,ij} = v_i \cdot v_j^T$. The above equation (10) can be converted to

$$\hat{y}_f = w_{u,0} + \sum_{i=1}^{d} w_{u,i} x_{f,i} + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle v_i, v_j \rangle x_{f,i} x_{f,j}, \tag{11}$$

where $\langle v_i, v_j \rangle = \sum_{l=1}^{e} v_{i,l} \cdot v_{j,l}$. The training parameters are integrated into $w_u = (w_{u,0}, \{w_{u,1}, w_{u,2}, \cdots, w_{u,d}\}, \{v_{1,1}, v_{1,2}, \cdots, v_{d,e}\})$, and the quadratic term learning parameters are $d * e$.

4.1.2. Training of the Local Model. To track the changes in user preferences and protect user privacy, we design a local model training process using user request records as the input data for model training. To measure the learning performance of the model, we use the cross-entropy loss function to represent the loss of UE$_u$ for the binary classification problem. The formula is as follows:

$$\mathrm{Loss}\left(w_u, x_f, y_f\right) = -y_f \log \hat{p}_{t,u,f} - \left(1 - y_f\right) \log \left(1 - \hat{p}_{t,u,f}\right). \tag{12}$$

When the user preference model update starts, we assume that UE$_u$ receives R_u requests. Based on the collected samples, we iteratively learn the user preference model parameters by minimizing the logistic loss for each sample, denoted as

$$w_u^{r+1} = \operatorname{argmin}\left(Loss\left(w_u, x_f, y_f\right)\right) r = 1, 2, \cdots, R_u, \tag{13}$$

where w_u^{r+1} is defined as the model parameters learned by UE$_u$ in the k-th iteration.

Due to the constructed dataset's extensive feature dimensionality and sparseness, overfitting may occur. Follow-the-regularized-leader (FTRL) [14] is used to solve Equation (14). FTRL is an online optimization method based on the online gradient descent (OGD) method, and Equation (15) is the iterative strategy of OGD. FTRL introduces both L1 and L2 mixed regularization terms into the optimization process. The L1 regularization term increases the sparsity of the model solution, and the L2 regularization term helps to prevent the model from overfitting. The update strategy of FTRL is

$$w_u^{r+1} = w_u^r - \eta_r g_r, r = 1, 2, \cdots, R_u, \tag{14}$$

$$w_u^{r+1} = \operatorname{argmin}\left(g_{1:r} \cdot w_u + \frac{1}{2}\sum_{s=1}^{r} \sigma_s \left\| w_u - w_{s,u} \right\|_2^2 + \lambda_1 \| w_u \|_1 + \frac{1}{2}\lambda_2 w_{u2}^{2}\right), r = 1, 2, \cdots, R_u, \tag{15}$$

where η_r denotes a nonincreasing learning rate. σ is a parameter related to η_r that satisfies $\sum_{s=1}^{r}\sigma_s = 1/\eta_r$. λ_1 and λ_2 denote regularization parameters with positive values, and $g_{1:r} = \sum_{s=1}^{r} g_s$ is the sum of the gradient vectors of the first r samples. The gradient vector g_r of the r-th sample is represented as follows

$$g_r = \nabla \mathrm{Loss}(w_u, x_k, y_k) = \begin{cases} \widehat{p}_u - y_r & w_u = w_{u,0}, \\ (\widehat{p}_u - y_r) x_{r,i} & w_u = w_{u,i}, \\ (\widehat{p}_u - y_r)\left(x_{r,i}\sum_{j=1}^{d} v_{j,l} x_{r,j} - v_{i,l} x_{r,i}^2\right) & w_u = v_{i,l}. \end{cases} \tag{16}$$

By continuing the expansion of Equation (16), we can obtain:

$$\begin{aligned} w_u^{r+1} &= \operatorname{argmin}\left(g_{1:r} \cdot w_u + \frac{1}{2}\sum_{s=1}^{r}\sigma_s\left(w_u{}^T w_u - 2 w_u{}^T w_{s,u} + w_{s,u}{}^T w_{s,u}\right) + \lambda_1 \|w_u\|_1 + \frac{1}{2}\lambda_2 \|w_u\|_2^2\right) \\ &= \operatorname{argmin}\left(\left(g_{1:r} - \sum_{s=1}^{r}\sigma_s\right)\cdot w_u + \frac{1}{2}\sum_{s=1}^{r}\sigma_s\left(w_{s,u}{}^T w_{s,u}\right) + \frac{1}{2}\left(\sum_{s=1}^{r}\sigma_s + \lambda_2\right)\left(w_u{}^T w_u\right) + \lambda_1\|w_u\|_1\right) \\ &= \operatorname{argmin}\left(z_r{}^T \cdot w_u + \frac{1}{2}\left(\sum_{s=1}^{r}\sigma_s + \lambda_2\right)\|w_u\|_2^2 + \lambda_1 \|w_u\|_1 + \frac{1}{2}\sum_{s=1}^{r}\sigma_s \cdot \|w_{s,u}\|_2^2\right) \end{aligned} \tag{17}$$

where $1/2\sum_{s=1}^{r}\sigma_s w_{s,u_2}^{\ 2}$ is a constant that does not affect the problem. Let $z_r = g_{1:r} - \sum_{s=1}^{r}\sigma_s w_{s,u}$; we can then obtain the z_r iteration relationship as follows:

$$z_r = z_{r-1} + g_r + \sigma_r w_{r,u} \tag{18}$$

For the requested content, there is a difference in the weight change rate of each feature dimension, and the gradient value of each feature dimension reflects this change rate. Therefore, different learning rates are used for different feature dimensions:

$$\begin{aligned} g_r &= \left(g_{u,0}^r, \left\{g_{u,1}^r, g_{u,2}^r, \cdots, g_{u,d}^r\right\}, \left\{g_{1,1}^r, g_{1,2}^r \cdots g_{d,e}^r\right\}\right)^T, \\ z_r &= \left(z_{u,0}^r, \left\{z_{u,1}^r, z_{u,2}^r, \cdots, z_{u,d}^r\right\}, \left\{z_{1,1}^r, z_{1,2}^r \cdots z_{d,e}^r\right\}\right)^T, \\ w_u &= \left(w_{u,0}, \left\{w_{u,1}, w_{u,2} \cdots w_{u,d}\right\}, \left\{v_{1,1}, v_{1,2} \cdots v_{d,e}\right\}\right)^T, \\ w_u^{r+1} &= \left(w_{u,0}^{r+1}, \left\{w_{u,1}^{r+1}, w_{u,2}^{r+1} \cdots w_{u,d}^{r+1}\right\}, v_{1,1}^{r+1}, v_{1,2}^{r+1} \cdots v_{d,e}^{r+1}\right)^T. \end{aligned} \tag{19}$$

The feature dimension is $N = 1 + d + d * e$, and the n-th feature learning rate can be denoted as $\eta_{r,n} = \alpha/(\beta + \sqrt{\sum_{s=1}^{r} g_{s,n}{}^2}) = \alpha/(\beta + \sqrt{q_{s,n}})$, where α and β are tuning parameters chosen to yield good learning performance. These two parameters are FTRL optimization parameters. Equation (18) can be split into subproblems for each feature

$$w_{u,n}^{r+1} = \operatorname{argmin}\left(z_{r,n} w_{u,n} + \lambda_1 \left|w_{u,n}\right| + \frac{1}{2}\left(\sum_{s=1}^{r}\sigma_s + \lambda_2\right)\left|w_{u,n}\right|^2\right) r = 1, 2, \cdots, R_u \tag{20}$$

where the L1 norm is nondifferentiable at $w_{u,n} = 0$. Define $\partial |w_{u,n}|$ as the subgradient of the L1 norm when $w_{u,n} = w_{r+1,u,n}$. The optimal solution should satisfy that the derivative is 0, so $z_{r,n} + \lambda_1 \partial |w_{u,n}| + 1/2(\sum_{s=1}^{r}\sigma_s + \lambda_2) w_{u,n} = 0$ is obtained by derivation of the above formula. Thus, we have:

$$w_u^{r+1} = \begin{cases} 0, & f\left|z_{r,n}\right| < \lambda \\ \dfrac{\lambda_1 \operatorname{sgn}(z_{r,n}) - z_{r,n}}{\lambda_2 + \left(\beta + \sqrt{q_{s,n}}\right)/\alpha}, & \text{otherwise} \end{cases} \quad r = 1, 2, \cdots, R_u. \tag{21}$$

4.1.3. Model Aggregation. After each UE completes the local model training process, to obtain the overall content popularity, BS_m must aggregate the models trained by all UEs, using the federated average method for aggregation. The parameters of the global model are formulated as:

$$w_m = \sum_{u=1}^{U_m^t} \frac{R_u \cdot w_u}{\sum_{u \in U_m^t} R_u}. \tag{22}$$

At this time, the regional popularity P_m^t of BS_m can be obtained by Formula (10). The time complexity of the proposed prediction model algorithm is $O(U_m^t R_u N)$, and the specific algorithm is shown in Algorithm 1.

4.2. Deep Reinforcement Learning-Based Content Caching Decision. After obtaining the content collection, it is

```
1: Input: $\alpha^u, \beta^u, \alpha^v, \beta^v, L_1^u, L_2^u, L_1^v, L_2^v$
2: Initialization: $w_u, z_u, q_u$
3: For $u = 1, 2, \cdots, U_m^t$ do
4:   For $r = 1, 2, \cdots, R_u$ do
5:     Calculate $g_{u,0}^r$ by (17)
6:     $\sigma_{u0}^r = (1/\alpha^u)(\sqrt{q_{u,0}^{r-1} + (g_{u,0}^{r-1})^2} - \sqrt{q_{u,0}^{r-1}})$
7:     $z_{u,0}^r = z_{u,0}^{r-1} + g_{u,0}^r + \sigma_{u,0}^r + w_{u,0}^r$
8:     $q_{u,0}^r = q_{u,0}^{r-1} + (g_{u,0}^r)^2$
9:     For $i \in d$ do
10:        Calculate $g_{u,0}^r$ by (17)
11:        $\sigma_{ui}^r = (1/\alpha^u)(\sqrt{q_{u,i}^{r-1} + (g_{u,i}^{r-1})^2} - \sqrt{q_{u,i}^{r-1}})$
12:        $z_{u,i}^r = z_{u,i}^{r-1} + g_{u,i}^r + \sigma_{u,i}^r + w_{u,i}^r$
13:        $q_{u,i}^r = q_{u,i}^{r-1} + (g_{u,i}^r)^2$
14:        For $l \in e$ do
15:          Calculate $g_{d,e}^r$ by (17)
16:          $\sigma_{il}^r = (1/\alpha^v)(\sqrt{q_{i,l}^{r-1} + (g_{i,l}^{r-1})^2} - \sqrt{q_{i,l}^{r-1}})$
17:          $z_{i,l}^r = z_{i,l}^{r-1} + g_{i,l}^r + \sigma_{i,l}^r + w_{i,l}^r$
18:          $q_{i,l}^r = q_{i,l}^{r-1} + (g_{i,l}^r)^2$
19:        End for
20:      End for
21:   End for
22: End for
23: Calculate $w_u$ by (21)
24: Calculate $w_m$ by (22)
25: Calculate $\widehat{p}_{t,u,f}$
```

Algorithm 1: The proposed content popularity prediction algorithm based on FL.

necessary to determine which content must be placed in the edge server to minimize the cost to the video provider. Since edge cache environments usually have huge high-dimensional state spaces, it is difficult to manually determine all valuable features from the environment. Deep reinforcement learning can automatically obtain the optimal policy from the original high-dimensional state input to solve such problems. DQN is a general DRL framework, but it often overestimates the Q value of the possible actions in a given state. Additionally, DQN usually estimates Q values for all actions of each state, but this is not necessary for those states where actions have no effect on the environment or Q values. Therefore, we propose a content placement method based on Dueling-DDQN, which combines double DQN and Dueling DQN to effectively reduce the overestimation of DQN and accelerate the learning process. The main purpose of double DQN is to mitigate the overestimation problem. Dueling DQN decomposes the action-value function into a state-value function and a dominance function to speed up convergence without estimating the Q value of each action in each state.

The method has the following three steps. (1) First, the cooperative content caching problem is formulated as a constrained Markov decision process (CMDP). (2) Second, the cache placement process is analyzed, and the reward function for the cache decision is constructed. (3) Finally, deep reinforcement learning is used to obtain the optimal content placement policy. The CM is considered a proxy in a given scenario, making caching decisions for all MEC servers. The CMDP element can be represented as a four-tuple consisting of (S, A, R, and C), where S is the state space, A is the action space, R is the reward, and C is the constraint. The detailed definition is as follows:

(1) State space: S denotes the set of edge cache node states, and $S^t = \{S_1^t, S_2^t, \cdots, S_M^t\}$ denotes the specific state of edge cache nodes in time slot t. The CM collects information such as content popularity vector and cache capacity of each edge base station

(2) Action space: action space A is defined as $A^t = \{D_1^t, D_2^t, \cdots, D_M^t\}$, where A^t is the action space set of time slot t and represents the buffering decisions of D_M^t in time slot t. The CM will select a^t from the action space as the buffer decision of the BS according to the information received from each base station

(3) Reward: $R(S^t, a^t)$ represents the reward obtained by a BS for performing action a^t in state S^t. From Formula (8), we can see that the optimization goal of this paper is to minimize the cost of the video provider. Therefore, $R(S^t, a^t)$, the profit gained from storing a single file, can be defined as:

$$R\left(S^t, a^t\right) = -\cos \mathrm{t}_f. \tag{23}$$

(4) Constraints: in making cache decisions, it is necessary to ensure that the files cached by each BS do not exceed its capacity. The capacity constraint is defined as follows:

$$\sum_{f=1}^{F} d_{m,f} \cdot z_f \le s_m. \tag{24}$$

The Dueling-DDQN algorithm is a deep neural network algorithm used to predict the size of the Q value. The Q value can be understood as the state action value, i.e., the expected benefit of the agent acting in a certain state. The algorithm divides the entire model structure into two parts: the state value function and the advantage function. The state value function is used to estimate the value of a state, while the advantage function is used to estimate the advantage of an action taken in a state. The value function can be expressed as:

$$Q\left(S^t, a^t, \theta\right) = V\left(S^t, \theta\right) + A\left(S^t, a^t, \theta\right). \tag{25}$$

It is impossible to obtain a unique determination of V and A based on a given value of Q. Therefore, Equation (25) is not identifiable. To solve this problem, a centralized treatment of the dominance function part Q function is given by

$$Q\left(S^t, a^t, \theta\right) = V\left(S^t, \theta\right) + \left(A\left(S^t, a^t, \theta\right) - \frac{1}{|A|}\sum_{a'} A\left(S^t, a', \theta\right)\right), \tag{26}$$

where $|A|$ denotes the dimensionality of the vector $A(S^t, a^t, \theta)$. The weight parameters of the target Q network must be updated once per cycle during the training process. The parameters of the updated network are updated by stochastic gradient descent (SGD) to minimize the loss function.

$$L_{\text{loss}} = E\left[\text{Target}tQ - Q\left(S^t, a^t, \theta\right)\right]. \tag{27}$$

The whole training process involves approximating the Q value to the target Q value, so the target Q value is expressed as:

$$\text{Target}Q = R\left(S^t, a^t\right) + \gamma Q\left(S', \operatorname{argmin}Q\left(S', a', \theta\right), \theta'\right), \tag{28}$$

where $\operatorname{argmin}Q(S', a', \theta)$ represents the action corresponding to the maximum Q value in the current Q network. The selected action is then used to calculate the target Q value in the target network. The detailed process of the caching strategy based on Dueling-DDQN is shown in Algorithm 2.

5. Simulation Results

In this section, the experimental results of the proposed algorithms are investigated, and the performance of four other algorithms, namely, content caching algorithm based on marginal gain, FL-based caching strategy, popularity-based caching algorithm, and noncooperative caching strategy, is taken as a reference.

5.1. Simulation Parameters. In the experiment, it is assumed that the number of base stations M is [3,6], and the capacity of base stations s_m is set to [100,300] MB identically. The number of users is 30, the users are randomly distributed under different BSs, and the content size z_f is set to [5,10] MB. The data rate between each BS m and the CM is set to 128 MB/s, and the data rate between the CM and cloud server c is set to 32 MB/s. The parameters used in the simulation experiment are shown in Table 3.

5.2. Datasets. To evaluate the performance of the proposed edge caching strategy, we use a real-world dataset-MovieLens [33]. The MovieLens ([https://grouplens.org/datasets/movielens/]) dataset contains rating data for multiple movies by multiple users, movie metadata information, and user attribute information. The MovieLens 100 K dataset contains 100,000 ratings for 1682 movies by 943 users. Each user has reviewed at least 20 movies with ratings on a 5-star scale, from 0 to 5. This paper simulates the process of users requesting content. We assume that the movie participation score is the content requested by the user, and each movie score corresponds to a content download. Literature [14, 28] take a similar approach to simulate the process of a user requesting content.

5.3. Performance Metrics. This paper considers three performance metrics: cache hit ratio (hit), average transmission delay (time), and cost. The cache hit ratio represents the ratio of satisfied requests to the total number of requests at the edge node. It is defined as:

$$\text{hit} = \frac{R_{\text{request}} - R_{\text{miss}}}{R_{\text{request}}}, \tag{29}$$

where R_{request} is the total number of requests received by the edge server in each time slot, and R_{miss} is the number of missed requests. The average transmission delay represents the average delay to transmit the content from the edge server or the central server to the user. It is expressed as:

$$\text{time} = \sum_{u=1}^{N_m^t}\sum_{f=1}^{F}\left(x_{u,m,f}^t \cdot \frac{z_f}{\phi_{u,m}} + x_{u,i,\mathrm{f}}^t \cdot \left(\frac{2z_f}{\phi_{c,m}} + \frac{z_f}{\phi_{u,m}}\right)\right) + x_{u,c,f}^t \cdot \frac{\left(\left(z_f/\phi_{c,o}\right) + \left(z_f/\phi_{c,m}\right) + \left(z_f/\phi_{u,m}\right)\right)}{R_{\text{request}}}, \tag{30}$$

```
1: Input: The capacity of the experience replay poll $N$, train starts $TR$, size of minibatch $K$, discount factor $\gamma$, $\varepsilon$-greedy exploration $\varepsilon$, learning rate $\alpha$, number of episodes $E$, target network update period $C$
2: Initialization: $w, w'$, TargerQ
3: For ep $= 1, 2, \cdots, E$ do
4:   Input initial state space $S_t$.
5:   For $t = 1, 2, \cdots$ do
6:     Choose an action $a_t$ via $\varepsilon$-greedy policy
7:     Execute action $a_t$ and get the next status $S_{t+1}$ and reward $R(S_t, a_t)$, judge whether $S_{t+1}$ is a terminal state.
8:     Put the sample $(S_t, a_t, R(S_t, a_t), S_{t+1})$ into the experience replay pool.
9:     If $t\%TR == 0$ then
10:        Randomly sample training samples with a minibatch size K from the experience replay pool $N$.
11:        Calculate the target $q$ value by formula (5)
12:        Apply the SGD method to calculate equation (6) to update the weight $w$
13:      End if
14:      If $t\%C == 0$ then
15:        Update target Q network parameters $w' = w$
16:      End if
17:      $S_t = S_{t+1}$
18:   End for
19: End for
```

Algorithm 2: The edge cache algorithm based on Dueling-DDQN.

Table 3: Key notations and values.

Notation	Description	Value
M	Number of BSs	[3,6]
F	Number of contents	1682
U	Number of users	[30,60]
s_m	Capacity of BSs	[100,300]MB
z_f	Content size	[8,16]MB
$\phi_{o,m}$	Data rate between each BS m and CM	128 MB/s
$\phi_{c,o}$	Data rate between CM and cloud server c	32 MB/s

where the above equation is the ratio of the transmission delay of all requests and the number of all requests between time slot t. The cost is the optimization objective function in this paper.

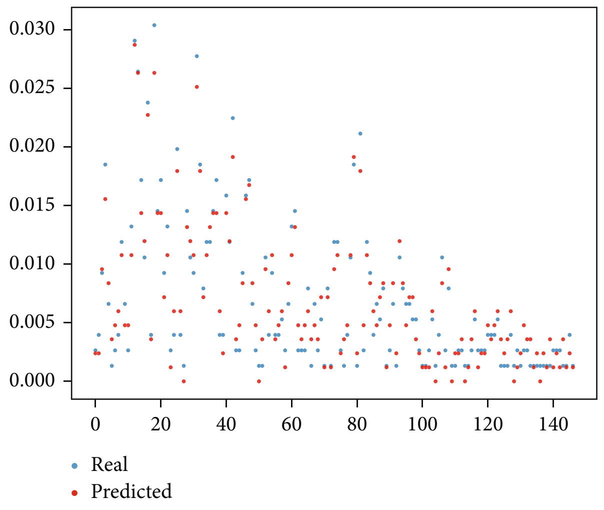


Figure 2: Comparison of the predicted popularity and the real popularity in the finite time horizon T.

5.4. Results. To evaluate the performance, the proposed algorithm is compared with the following four algorithms:

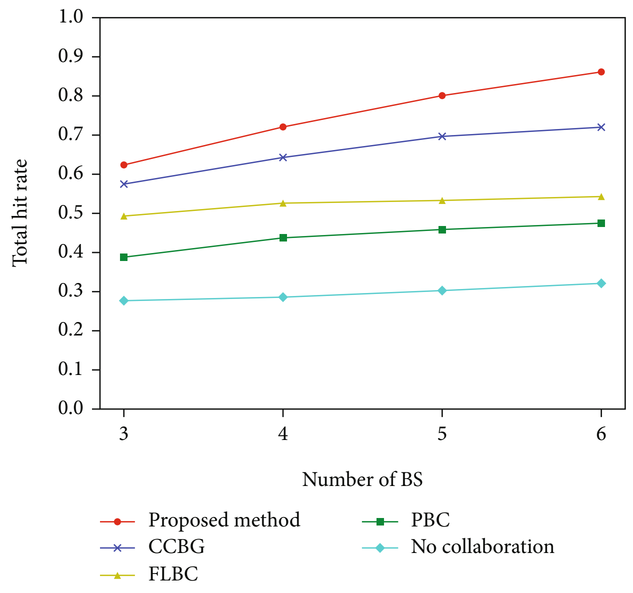
(1) Content caching strategy based on marginal gain [30](CCBG): this strategy analyzes the marginal gain in latency and cache cost to cache the content in the edge server

(2) FL-based caching policy [28](FLBC): this algorithm applies FL to user demand prediction and formulates the caching problem as an integer linear programming (ILP) problem

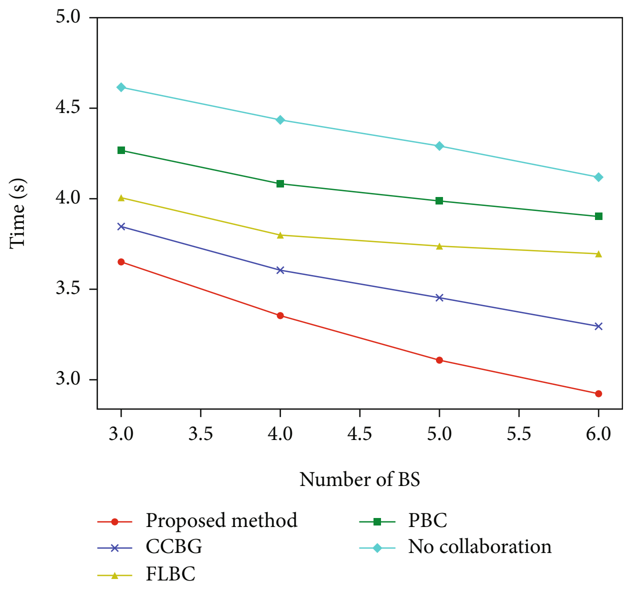
(3) Popularity-based caching algorithm [34](PBC): this is a caching update scheme based on content popularity, replacing low-popularity content with high-popularity content

(4) Noncooperative caching strategy (no collaboration): this algorithm does not consider cooperation between base stations for content storage

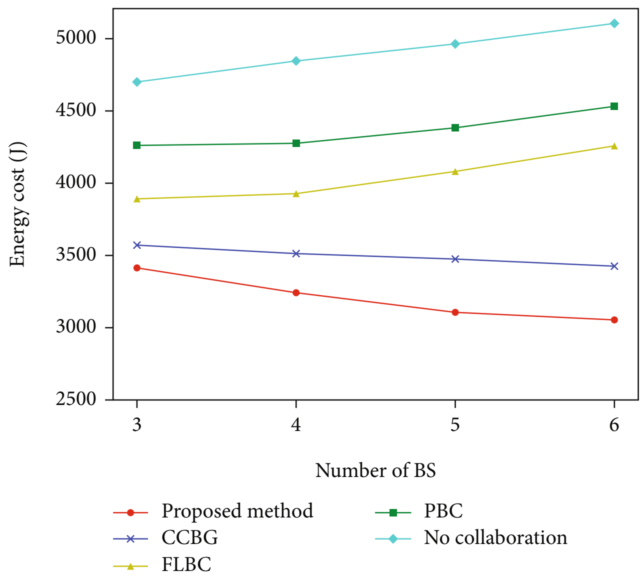
In Figure 2, the predicted popularity is shown using the proposed request prediction strategy. The predicted popularity is very close to the actual popularity, which indicates



(a) Effect of the number of BSs on hit rate



(b) Effect of the number of BSs on average latency

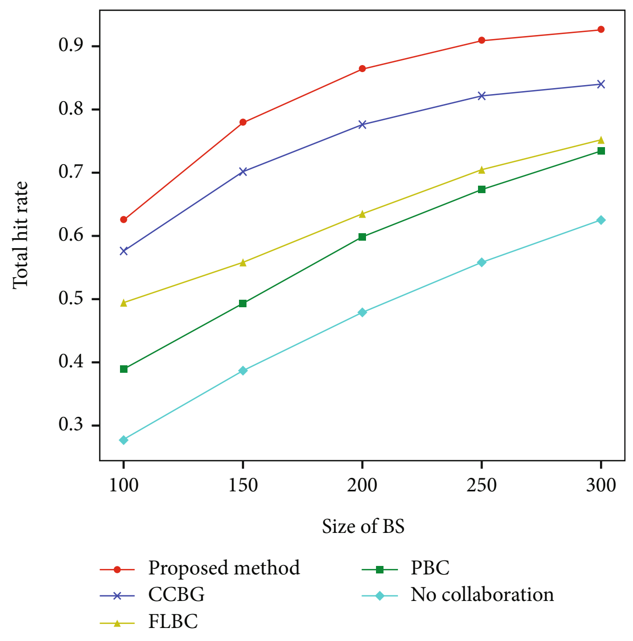


(c) Effect of the number of BSs on energy cost

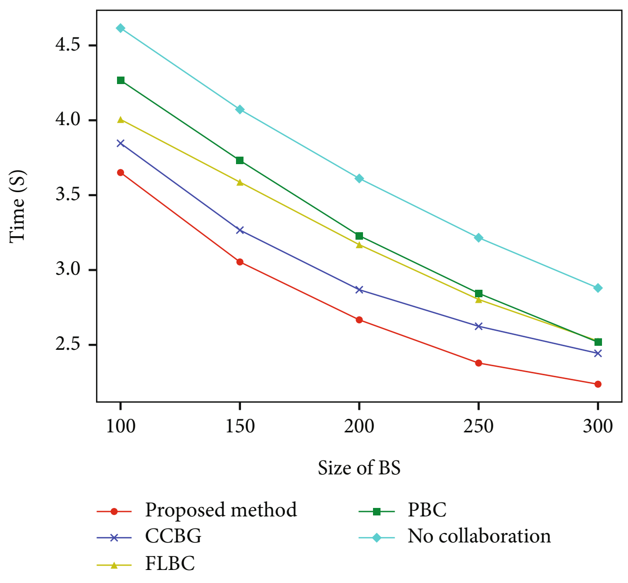
Figure 3: Effect of the number of BSs on cache performance.

that our proposed popularity prediction strategy can effectively predict user requests.

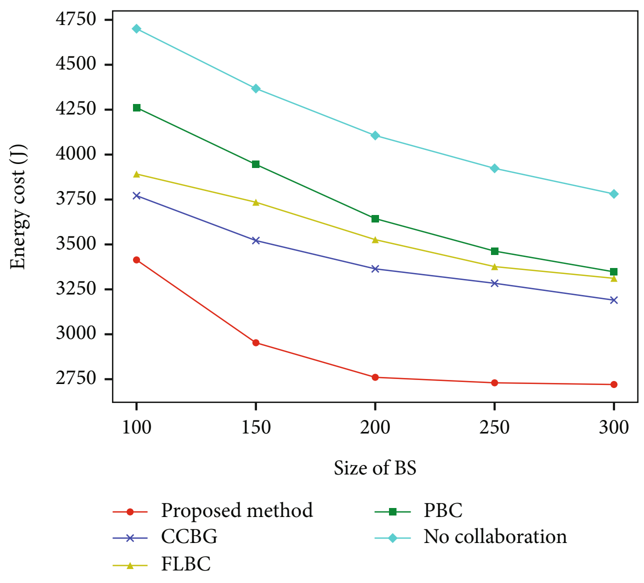
Figure 3 shows the impact of the number of BSs on the cache hit rate, average transmission delay, and cost. The capacity of BSs is set to 100 MB, and the number of BSs is increased from 3 to 6. From Figures 3(a) and 3(b), it can be seen that the cache hit ratio increases and the average transfer latency decreases for the four algorithms as the number of BSs increases. In addition, the proposed algorithm consistently outperforms the four baseline algorithms. The reason is that as the number of BSs increases, the amount of content that the collaborative baseline can cache increases; however, the number of users in the collaborative range also increases, and the types of content requests from users become more diverse. This scenario leads to a gradual slowing of the hit rate variation and a gradual decrease in the delay variation. The increase in the number of users leads to an increase in the number of requests, so the hit rate and latency still show an improving trend. From Figure 3(c), the comparison shows that the proposed service placement method outperforms the other cache placement methods. The overall cost of the comparison algorithm is increasing, and the total cost of the proposed algorithm is decreasing. There is a downward trend in transport costs as most requests can be responded to locally. However, as the number of BSs increases, storage costs also increase. The proposed algorithm is more optimized for the transfer cost.



(a) Effect of the caching capacity for each BS on hit rate



(b) Effect of the caching capacity for each BS on average latency

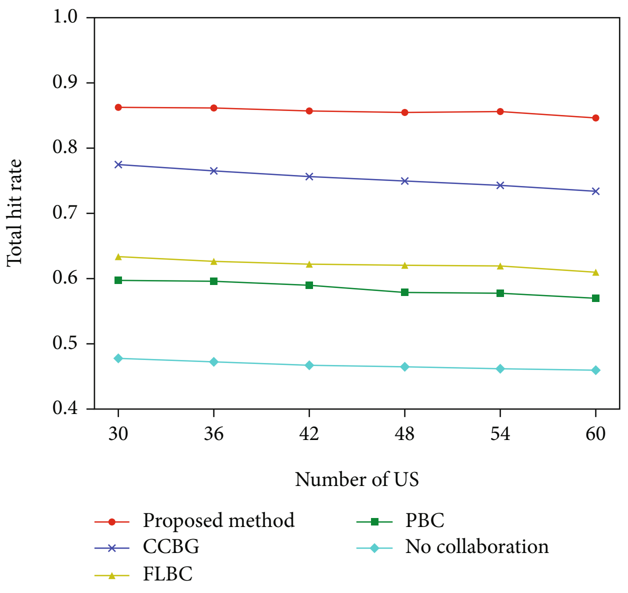


(c) Effect of the caching capacity for each BS on energy cost

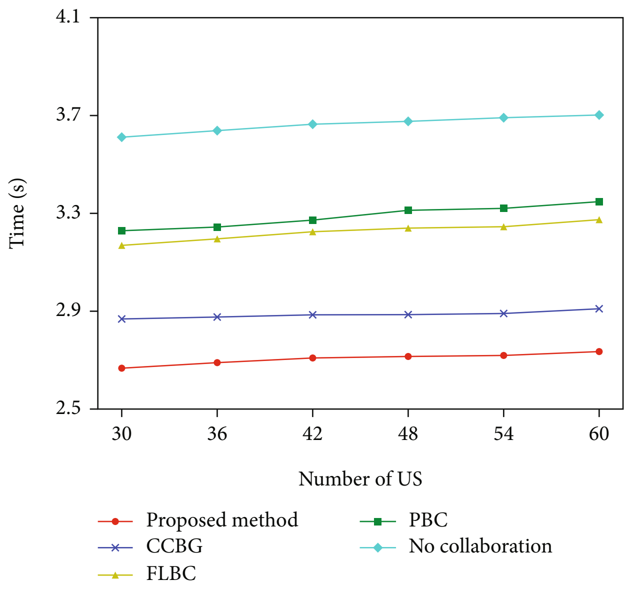
Figure 4: Effect of the caching capacity for each BS on cache performance.

Figure 4 shows the impact of the BS capacity on the cache hit ratio, average transmission delay, and cost. The number of BSs is set to 3, and the capacity of the BSs is set to 100 MB-300 MB. As shown in Figure 4(a), as the storage capacity of the BSs increases, the cache hit rate gradually increases. That is because as the cache capacity of the BSs increases, the BS can cache more files, so the cache hit rate increases. As shown in Figure 4(b), as the storage capacity of the BSs increases, the delay gradually decreases the number of times files are obtained from the cloud server decreases, so the delay decreases. As shown in Figure 4(c), as the BS storage capacity increases, the video provider's cost decreases. Because more content can be requested locally without going through the cloud center, the content with the most significant profit gain will be cached first as the cache capacity increases. Hence, the hit rate, latency, and profit curve tend to change faster initially and then more slowly.

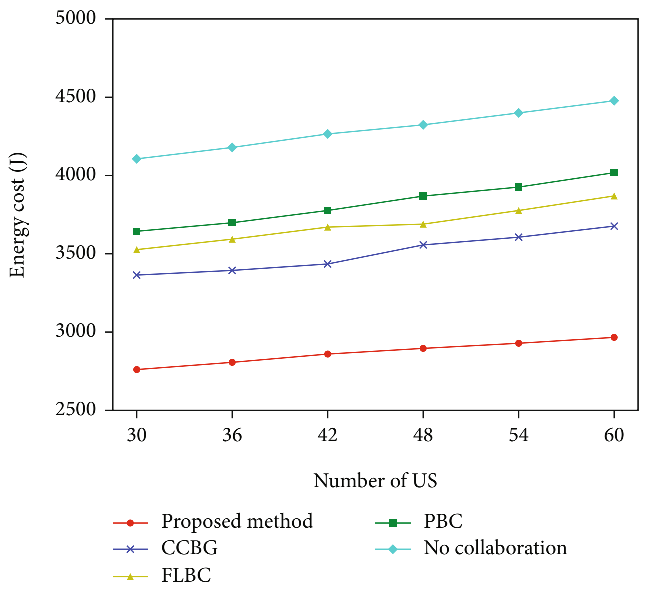
Figure 5 shows the impact of the UE number on the cache hit rate, average transmission delay, and cost. The number of BSs is set to 3, the capacity of the BSs is set to 200 MB, and the number of UEs served is increased from 30 to 60. As shown in Figure 5(a), as the number of UEs increases, the cache hit rate shows a downward trend. This is because as the number of UEs increases, the content requested by users becomes diverse, and content requests



(a) Effect of the number of UEs on hit rate



(b) Effect of the number of UEs on average latency



(c) Effect of the number of UEs on energy cost

Figure 5: Effect of the number of UEs on cache performance.

become scattered, so the hit rate decreases accordingly. As shown in Figure 5(b), the delay increases gradually as the number of UEs increases. It is because as the cache hit rate of the BSs decreases, the number of times files are obtained from the cloud server increases, so the delay also decreases. As shown in Figure 5(c), the video provider cost increases as the number of UEs increases. Because the number of requests passing through the cloud server increases, the overall cost of video providers increases.

6. Conclusion

In this paper, we propose a deep reinforcement learning-based approach to collaborative content caching to optimize video providers' costs. First, the content caching problem is represented as a CMDP. Then, the content caching process is analyzed to construct a caching reward function. Finally, deep Q-learning is used to obtain the optimal content caching strategy. In addition, considering the user's content request and privacy security, federated learning is used to design the caching strategy to make distributed predictions for the users in the nodes. Simulation results based on real datasets show that the proposed algorithm optimizes the cost of the video provider while achieving a high cache hit rate.

Although the cooperative caching strategy proposed in this paper has achieved good results, there are still some shortcomings. The proposed algorithm is a centralized algorithm and may not be suitable for network scenarios with a large number of base stations. Future work will aim at designing a scheme for multiagent proxy DRL that can learn the optimal caching policy where each BS acts as an agent and makes its own caching decisions.

Data Availability

Data is openly available in a public repository. The data that support the findings of this study are openly available in [movielens] at [https://grouplens.org/datasets/movielens/].

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 61902112, and 62072159.

References

[1] R. Aghazadeh, A. Shahidinejad, and M. Ghobaei-Arani, “Proactive content caching in edge computing environment: a review,” *Software: Practice and Experience*, 2021.

[2] H. Wu, Y. Fan, Y. Wang, H. Ma, and L. Xing, “A comprehensive review on edge caching from the perspective of total process: placement, policy and delivery,” *Sensors*, vol. 21, no. 15, p. 5033, 2021.

[3] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, “Online collaborative data caching in edge computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 281–294, 2021.

[4] B. N. Bharath, K. G. Nagananda, D. Gündüz, and H. V. Poor, “Caching with time-varying popularity profiles: a learning-theoretic perspective,” *IEEE Transactions on Communications*, vol. 66, no. 9, pp. 3837–3847, 2018.

[5] E. K. Markakis, K. Karras, A. Sideris, G. Alexiou, and E. Pallis, “Computing, caching, and communication at the edge: the cornerstone for building a versatile 5G ecosystem,” *IEEE Communications Magazine*, vol. 55, no. 11, pp. 152–157, 2017.

[6] A. Ioannou and S. Weber, “A survey of caching policies and forwarding mechanisms in information-centric networking,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2847–2886, 2016.

[7] Y. Jiang, Y. Hu, M. Bennis, F.-C. Zheng, and X. You, “A mean field game-based distributed edge caching in fog radio access networks,” *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1567–1580, 2019.

[8] P. Ray, R. Kaluri, T. Reddy, and K. Lakshmanna, “Contemporary developments and technologies in deep learning–based IoT,” in *Deep Learning for Internet of Things Infrastructure*, pp. 61–82, CRC Press, 2021.

[9] T. Shelatkar, D. Urvashi, M. Shorfuzzaman, A. Alsufyani, and K. Lakshmanna, “Diagnosis of brain tumor using light weight deep learning model with fine-tuning approach,” *Computational and Mathematical Methods in Medicine*, vol. 2022, Article ID 2858845, 9 pages, 2022.

[10] K. Qi, S. Han, and C. Yang, “Learning a hybrid proactive and reactive caching policy in wireless edge under dynamic popularity,” *IEEE Access*, vol. 7, pp. 120788–120801, 2019.

[11] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, “Content popularity prediction and caching for ICN: a deep learning approach with SDN,” *IEEE access*, vol. 6, pp. 5075–5089, 2017.

[12] H. Feng, Y. Jiang, D. Niyato, F.-C. Zheng, and X. You, “Content popularity prediction via deep learning in cache-enabled fog radio access networks,” in *2019 IEEE global communications conference (GLOBECOM)*, pp. 1–6, Waikoloa, HI, USA, 2019.

[13] Q. Fan, X. Li, J. Li, Q. He, K. Wang, and J. Wen, “PA-cache: evolving learning-based popularity-aware content caching in edge networks,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1746–1757, 2021.

[14] Y. Jiang, M. Ma, M. Bennis, F.-C. Zheng, and X. You, “User preference learning-based edge caching for fog radio access network,” *IEEE Transactions on Communications*, vol. 67, no. 2, pp. 1268–1283, 2018.

[15] Z. Yu, J. Hu, G. Min et al., “Federated learning based proactive content caching in edge computing,” in *2018 IEEE global communications conference (GLOBECOM)*, pp. 1–6, Abu Dhabi, United Arab Emirates, 2018.

[16] K. Qi and C. Yang, “Popularity prediction with federated learning for proactive caching at wireless edge,” in *2020 IEEE wireless communications and networking conference (WCNC)*, pp. 1–6, Seoul, Korea (South)., 2020.

[17] Y. Wu, Y. Jiang, M. Bennis, F. Zheng, X. Gao, and X. You, “Content popularity prediction in fog radio access networks: a federated learning based approach,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, Dublin, Ireland, 2020.

[18] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, “A provably efficient online collaborative caching algorithm for multicell-coordinated systems,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1863–1876, 2015.

[19] L. Wang, J. Li, M. Chen, S. Tang, and B. Zheng, “An incentive caching mechanism in wireless networks based on stackelberg game,” in *In 2019 IEEE International Conference on Consumer Electronics-Taiwan (ICCETW)*, Yilan, Taiwan, 2019.

[20] Y. K. Tun, A. Ndikumana, S. R. Pandey, Z. Han, and C. S. Hong, “Joint radio resource allocation and content caching in heterogeneous virtualized wireless networks,” *IEEE Access*, vol. 8, pp. 36764–36775, 2020.

[21] C. Zhong, M. C. Gursoy, and S. Velipasalar, “Deep reinforcement learning-based edge caching in wireless networks,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 48–61, 2020.

[22] W. Jiang, G. Feng, and S. Qin, “Optimal cooperative content caching and delivery policy for heterogeneous cellular networks,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1382–1393, 2016.

[23] S. Sun, W. Jiang, G. Feng, S. Qin, and Y. Yuan, “Cooperative caching with content popularity prediction for mobile edge caching,” *Tehnički vjesnik*, vol. 26, no. 2, pp. 503–509, 2019.

[24] L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, “Online proactive caching in mobile edge computing using bidirectional

deep recurrent neural network," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5520–5530, 2019.

[25] C. Li, L. Zhu, W. Li, and Y. Luo, "Joint edge caching and dynamic service migration in SDN based mobile edge computing," *Journal of Network and Computer Applications*, vol. 177, article 102966, 2021.

[26] F. Zhang, G. Han, L. Liu, M. Martínez-García, and Y. Peng, "Joint optimization of cooperative edge caching and radio resource allocation in 5g-enabled massive iot networks," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 14156–14170, 2021.

[27] Y. Liu, Z. Ma, Z. Yan, Z. Wang, X. Liu, and J. Ma, "Privacy-preserving federated _k_ -means for proactive caching in next generation cellular networks," *Information Sciences*, vol. 521, pp. 14–31, 2020.

[28] T. Xiao, T. Cui, S. R. Islam, and Q. Chen, "Joint content placement and storage allocation based on federated learning in f-rans," *Sensors*, vol. 21, no. 1, p. 215, 2021.

[29] F. Jiang, W. Cheng, Y. Gao, and C. Sun, "Caching strategy based on content popularity prediction using federated learning for f-ran," in *2021 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, pp. 19–24, Xiamen, China, 2021.

[30] C. Li, Y. Zhang, M. Song, X. Yan, and Y. Luo, "An optimized content caching strategy for video stream in edge-cloud environment," *Journal of Network and Computer Applications*, vol. 191, article 103158, 2021.

[31] C. Li, Y. Zhang, Q. Sun, and Y. Luo, "Collaborative caching strategy based on optimization of latency and energy consumption in MEC," *Knowledge-Based Systems*, vol. 233, article 107523, 2021.

[32] S. Rendle, "Factorization machines," in *2010 IEEE international conference on data mining*, pp. 995–1000, Sydney, NSW, Australia, 2010.

[33] F. M. Harper and J. A. Konstan, "The movielens datasets," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2016.

[34] H. Nakayama, S. Ata, and I. Oka, "Caching algorithm for content-oriented networks using prediction of popularity of contents," in *2015 IFIP/IEEE international symposium on integrated network management (IM)*, pp. 1171–1176, Ottawa, ON, Canada, 2015.