Received November 18, 2019, accepted December 1, 2019, date of publication January 7, 2020, date of current version January 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964711

Mobile Edge Cache Strategy Based on Neural Collaborative Filtering

YU CHEN, YONG LIU, JINGYA ZHAO, AND QINGHUA ZHU

School of Telecommunication Engineering, Beijing Polytechnic, Beijing 100176, China

Corresponding author: Yu Chen (buptchen@163.com)

This work was supported by the Beijing City Board of Education project under Grant KM202010858005.

ABSTRACT In order to effectively reduce the network transmission delay and improve the network transmission quality, the concept of Content Delivery Network (CDN) is brought forth to provide necessary technical support. In this paper, the edge cooperative caching (ECC) based on machine learning and greedy algorithm is put forward. To start with, the neural collaborative filtering is used to design the content popularity prediction algorithm to realize more accurate prediction of content popularity. Following that, the greedy algorithm after optimization is used to obtain the content delivery strategy of various servers in the cooperative cache domain. Finally, the ECC is adopted to achieve the optimization goal of minimal average content transmission delay. Meanwhile, the simulation experiment is carried out to verify the performance of the ECC. The experimental results suggest that the ECC can effectively improve the cache hit rate and the content cache space utilization, and shorten the average content transmission delay.

INDEX TERMS Collaborative filtering, neural network, content distribution network, mobile edge caching, proactive caching.

I. INTRODUCTION

With the explosive growth of data and the deepening interconnection of things of all forms, the data growth has far overtaken the growth of network bandwidth. Therefore, to simply maintain the original network computing method or to constantly improve the bandwidth can neither meet current users' requirement of network computing capacity [1], [2]. At the same time, technological advances of the current era have led to the emergence of many new applications, such as intelligent manufacturing and driverless technology, all of which are demanding about the data transmission and processing efficiency.

Meanwhile, the rapid development of intelligent terminals has brought about a sharp increase to the number of mobile devices. The most obvious case in point is the popularization of smartphones. In response to so many challenges, how to efficiently lower the transmission delay and improve the network transmission quality has become an issue of great concern, and the construction and edge computing of Content Delivery Network (CDN) have provided an effective solution plan [3,4]. As a special network that is different from the traditional network, CDN relies on edge servers distributed in different places to realize scheduling and load balancing of the content, thus ensuring efficient delivery of the content. Generally, network traffic chiefly depends on some files repeatedly downloaded from remote servers (such as Weibo videos and news reports which are prevailing on the network on a real-time basis), and the highly concentrated and repeated requests will impose tremendous loading pressure on the backhaul link of the network [5]. In order to cope with the enormous growth of network traffic, this paper makes use of the edge caching strategy, that is, to have the content with a high degree of popularity, such as video files and image files, cached in the nearer edge caching units, which usually refer to edge caching small base stations, before the use of network traffic reaches a peak. An advantage of the caching approach is that many users can acquire the popular content requested by them from the edge caching units rather than from the cloud data centers, thus largely cutting the expenses of the backhaul link and the use of network bandwidth. In this way, not only can the user experience be improved, but also the network space can be saved for other services so that the network can provide more services [6].

II. NETWORK MODEL

The edge caching algorithm and the corresponding Content Delivery Network (CDN) [7] can be designed based on

The associate editor coordinating the review of this manuscript and approving it for publication was Dapeng Wu.

the future popularity prediction. Before the network traffic reaches a peak, the popular content can be cached in the local server in advance to effectively bring down the redundant network transmissions and improve the network service performance [8]. At the same time, due to the limited storage of a server, the number of its users and content files cached in the local server is also limited. Concerning this problem, this paper designs an optimized model based on the edge cooperative caching, with multiple edge servers forming a cooperative caching unit. Every cooperative caching unit can be connected to the cloud data center, and every server in the cooperative cache domain is jointly responsible for users of the same region (See Fig. 1.) and reasonably distributing the content to every joint caching unit according to CDA. This can not only effectively reduce the redundant cache but also improve the cache space utilization rate of the cooperative caching unit to maximize the service performance [9], [10].

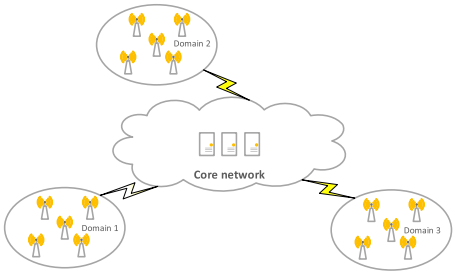

FIGURE 1. Multiple cooperative cache domain scenarios based on small base station cooperative caching system.

III. ALGORITHM ESTABLISHMENT

This paper assumes that M cache servers are deployed in a cooperative cache domain to realize the function of cooperative caching. The cooperative cache domain is named $C = \{c_1, c_2, \cdots c_M\}$. Every server is deployed at sites of different base stations for the convenient access of users in different regions. Assume that the cloud data center provides S pieces of different content, with the length of all content normalized to be L and the content set to be $O = \{o_1, o_2, \ldots, o_S\}$. Then, every requested content, o_i, of users abides by the rule of $o_i \in O, i = 1, 2, \ldots, S$. In order to better realize the prediction algorithm, this paper assumes that, in a unit time, the popularity of the content requested by users of a fixed region is unchanged, and that users' content request also remains unchanged. Assume that the static popularity is $P_{m,i}$ (denoting the degree of popularity of the content, o_i, within the service domain of the server, c_m). Following that, this paper defines a content caching matrix, $X = \{x_{m,i} | c_m \in C, o_i \in O\}$, (where o_i has been cached to the server, c_m; on the contrary, $x_{m,i} = 0$, meaning the content, o_i, is cached to the server, c_m), so as to clarify whether the popular content is cached in the cache domain and where the popular content is cached.

The content caching matrix per unit time should be updated. The superiority of this algorithm can be judged by the expenses of the network transmission. In such a system, there are multiple transmission expenses, including transmission expenses of sending the already cached content to users via the edge server, transmission expenses between edge cooperative cache servers, expenses for the server to have the popular content cached in the cloud data center, and expenses for users to directly acquire data expenses from the cloud data content. Under the preconceived conditions, users can request content from the server, c_m, in the cooperative cache server. This paper concentrates on the caching performance, so the expenses for users to directly acquire content from the cloud data center, and the expenses to send the content from the requested server to users can be ignored. Therefore, the final transmission expenses of this paper are decided by the sum of the transmission expenses between the edge cache servers and the transmission expenses of the remote cloud data center and the local server.

This paper assumes the transmission delay of every edge server to cache content in the cloud data center is $T_{m,0}$; $T_{m,n}$ is the unit transmission delay between any two adjacent cache servers, c_m and c_n, (two servers which can transmit content via the single-hop router); $d_{m,n}$ is the hop count of the router between two servers, c_m and c_n. Therefore, the transmission delay between two cache servers is $T_{m,n} \times d_{m,n}$. To cache content from the remote cloud data center to the local server will obviously incur a higher expense. Hence, it is obvious that $T_{m,0} > T_{m,n}$.

As to the transmission delay computing model, it is supposed that there is a user requesting the content o_i from one edge cache server c_m. Under the condition, there are three types of transmission delays. First, the content o_i has been cached to the user-requested server, c_m, which enables users to directly acquire the content. Second, the content o_i has been cached to c_n of the local cooperative caching unit, which cannot be directly accessed by the user, so the server c_n first delivers content to the server c_m and then delivers it for user access. Third, the content o_i is not in any local cooperative caching unit, so the server c_m should acquire the content from the cloud data center and then delivers it to the user. The three different types of transmission delays caused by user access are presented in Fig. 2 below:

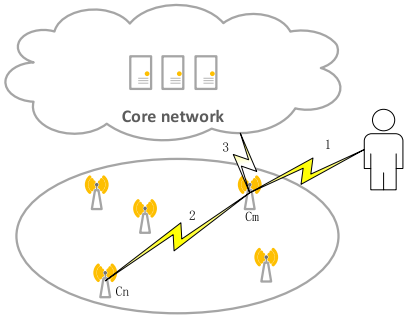

FIGURE 2. Three different types of transmission delays caused by user access.

According to the above discussions, every transmission type finally has the server c_m entrusting the content to the

user. Therefore, that c_m transmits the content to the user's service terminal is the time delay of every transmission type. Thus, the transmission delay is not included in the computing of expenses to measure the performance. Assume that the expenses incurred by the second type of transmission delay is $Y^1_{m,i}$, and that the expenses incurred by the third type of transmission delay is $Y^2_{m,i}$. Then, the following computing formula (1) can be obtained:

$$\begin{cases} Y^1_{m,i} = T_{m,n} \cdot L \cdot d_{m,k} \cdot (1 - x_{m,i}) \cdot [1 - \prod_{n \neq m}^{M=1} (1 - x_{n,i})] \\ Y^2_{m,i} = T_{m,0} \cdot L \cdot \prod_{n=1}^{M} (1 - x_{n,i}) \end{cases} \tag{1}$$

In (1), $d_{m,k} = \min_{n \neq m}\{d_{m,n} | x_{i,n} = 1\}$ denotes the hop count from the nearest c_n to c_m when there is no content cached in the edge server c_m, thus necessitating content to be acquired from the cooperative cache server. Where, the algebraic expression of $Y^1_{m,i}$ is not zero when the cached content is not in the local server, c_m, but in the remaining cache servers. Under the condition, $Y^1_{m,i}$ exists; otherwise, it does not exist. Similarly, the algebraic expression of $Y^2_{m,i}$ is not zero when the requested content is not cached in all servers. Then, the time delay, $Y^2_{m,i}$, exists. In this way, the above two expense computing formulas can be obtained. The objective of this paper is to improve the above two transmission expenses. Obviously, the cached content of a server has its upper limit. This paper assumes the upper limit for the content to be cached by an edge server is U_{m}. Thus, a constraint condition can be obtained, namely $\sum_{i=1}^{S} x_{m,i} \cdot L \leq U_m$.

Based on the above assumption, the optimized model can be obtained:

Optimization objective:

$$\min \sum_{m=1}^{M} \sum_{i=1}^{S} p_{m,i} \cdot (Y^1_{m,i} + Y^2_{m,i}) \tag{2}$$

Constraint:

$$\begin{aligned} &\sum_{i=1}^{S} x_{m,i} \cdot L \leq U_m, \quad \forall c_m \in C \\ &d_{m,k} = \min_{n \neq m}\{d_{m,n} | x_{i,n} = 1\}, \quad \forall c_m \in C \\ &x_{m,i} \in \{0, 1\}, \quad \forall c_m \in C, o_i \in O \end{aligned} \tag{3}$$

IV. SOLUTION OF THE OPTIMIZATION PROBLEM

To solve the optimization problem, this paper should first predict the future content popularity, $P_{m,i}$, and then adopt it as the input to obtain the predicted content caching matrix, $X = \{x_{m,i} | c_m \in C, o_i \in O\}$. Following that, the average transmission delay can be computed using the content caching matrix predicted by this paper. This paper divides the above process into two steps. Step 1 is to predict the future content popularity considering the historical popularity. Step 2 is to acquire the most suitable content for delivery according to the predicted value of the future content popularity and the goal of the optimized model. The flow chart is shown in Fig. 3 below:

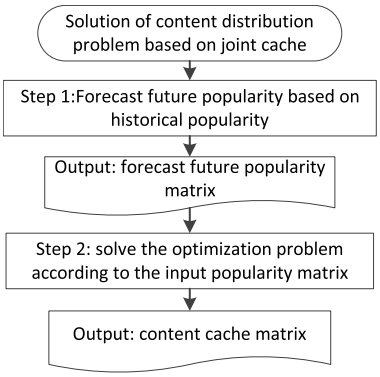

FIGURE 3. Flow chart of content delivery problem solving based on cooperative caching.

A. CONTENT POPULARITY PREDICTION BASED ON NEURAL COLLABORATIVE FILTERING

Content popularity refers to the degree of popularity of the content in a fixed region, which is a very important parameter to the establishment and design of the edge caching mechanism. Generally, the content popularity obeys the *Zipf* distribution, which is reflected as the ''80/20 Principle'', meaning that 80% of the popularity concentrates in 20% of the content. The popularity of a content file can be defined as the ratio of its user access to user access of all files. As a fast-changing data with a strong punctuality, content popularity is not easy to predict in the real world. Therefore, this paper assumes that the popularity within a cooperative cache domain remains the same within a unit time, which is known as the static popularity, $P_{m,i}$. According to the timeliness of popularity, the historical data of the static popularity over two continuously units of time are adopted as the training datasets to build the prediction model, which can obtain the future static popularity.

Of special note is that the popularity distribution of the total files in a region can be assumed to obey the *Zipf* distribution. However, due to differences of geographical positions of every independent server, the users show different preferences for the popular content. As a result, the popularity distribution of every independent server is usually different and might not obey the regional content popularity distribution. Therefore, this paper, in predicting the popularity, should consider and compute every server separately.

In the process of predicting the popularity, this paper makes use of machine learning and the neural collaborative filtering [11] to design the content popularity prediction strategy.

Specifically, the neural network-based neural collaborative filtering (NCF) framework [11] is employed to predict the probability of each user requesting one certain content. The NCF model consists of two main components: a generalized matrix factorization (GMF) module with linear core, and a multilayer perceptron (MLP) module with nonlinear core [12]–[14]. By combining these two kinds of modules, the model is able to achieve better accuracy performance in

predicting the user's preference for the contents. the NCF model is trained based on history request of users.

In more details, the user-content interaction is modelled by a multi-layer neural network. The output of each layer is the input of the next layer. The bottom level of input consists of two feature vectors that describe the user and the content, respectively. Above the input layer is the embedding layer, which is a fully connected layer and maps the sparse feature representation to a low-dimensional space vector. Then the results of the embedding layer are fed back to the multi-layer neural network, which is called a neural collaborative filtering layer. Anther the NCF layer, the potential feature vectors of the user and content are mapped into predictive scores. As long as the predictive score is obtained, the popularity of each content, i.e., $P_{m,i}$, can be calculated by summating the corresponding score over all the users served by the cache server m.

The specific algorithm is shown in literature [11]. In that literature in order to make the convergence model more flexible, the GMF and MLP are allowed to learn separately using their respective embedding layers and the last hidden layers of the two models are connected. This new model is named NeuMF in this paper. The specific method is as follows:

In (4), p_u^G,p_u^M represent the embedded layer output of user characteristics in the GMF and MLP, respectively, q_i^G, q_i^M represent the embedding layer outputs of the content features in the GMF and MLP, respectively.

$$\begin{aligned}\phi^{GMF} &= p_u^G \otimes q_i^G \\ \phi^{MLP} &= a_L\left(w_L^T\left(a_{L-1}\left(w_{L-1}^T\left(\ldots a_2\left(w_2^T\begin{bmatrix}p_u^M\\ q_i^M\end{bmatrix}\right)\right.\right.\right.\right. \\ &\qquad \left.\left.\left.\left.+b_2\right)+b_{L-1}\right)\right)+b_L\right) \\ \hat{y}_{ui} &= \sigma\left(h^T\begin{bmatrix}\phi_{GMF}\\ \phi_{MLP}\end{bmatrix}\right)\end{aligned} \tag{4}$$

B. PRE-TRAINING

Since the objective function of NeuMF model is non-convexity, gradient based optimization of network parameters can only find local optimal solutions. Studies in [15] have shown that the initialization process has an important impact on the convergence and performance of the deep learning model. Therefore, the initialization parameter can be set near the global optimal solution instead of using the random initialization method. So, the global optimal solution can be directly found by the gradient-based optimization method. Because the NeuMF model consists of GMF and MLP models, the GMF and MLP models are used for pre-training. First, the model parameters of GMF and MLP are randomly initialized and trained to converge. Then the corresponding portions of the NeuMF parameters are initialized using above model parameters. The only adjustment is in the output layer. The weights of the two models are combined

$$h \leftarrow \begin{bmatrix}\alpha h^{GMF}\\ (1-\alpha)h^{MLP}\end{bmatrix} \tag{5}$$

where h^{GMF}, h^{MLP} represent the parameter results obtained by pre-training the GMF and MLP models, respectively, and α is a hyperparameter used to weigh the impact of two pre-training models on the final result. For the pre-training of GMF and MLP, the adaptive moment estimation (Adam) is used, which adapts the learning rate of each parameter by performing small updates on frequent parameters and large updates on infrequent parameters. The Adam method converges faster on both models than the normal SGD and reduces the burden of adjusting the learning rate during training. After entering the pre-trained parameters into NeuMF, SGD instead of Adam is used to optimize the model. Because Adam needs to save the momentum information to properly update the parameters. Since NeuMF is initialized with pre-trained model parameters and no momentum information, it is not appropriate to further optimize NeuMF with a momentum-based approach.

C. PROACTIVE CACHING STRATEGY

In this section, the problem of backhaul link offloading in microcellular networks (SCNS) is addressed, where proactive caching plays a critical role. In this paper, the SBS deploys high-capacity caching devices, but the network has limited backhaul link capacity. On the basis of the aforementioned analysis, a proactive caching process is proposed to store contents according to the predicted popularity until the storage capacity is reached. The SBS stores a more accurate and popularity prediction result matrix. Each row of the matrix represents a user, and each column represents a content. If the user has requested a certain content, the interaction history between the user and the content is filled. However, the popularity matrix is highly sparse and partially unknown in practice. Therefore, the deep learning and collaborative filtering (CF) method is used and a distributed proactive caching process is proposed, using correlation of user - content to infer the probability of the user requests the content. For the contents which the user has never interacted with, the probability of interaction is predicted by collaborative filtering algorithm.

The mobile edge computing will make a judgment on the load of backhaul link. If the backhaul link is idle, the SBS will send a request in turn, and cache the top contents in the cache order table in advance to achieve the purpose of reducing the backhaul link load during peak hours.

First, users in the dataset are randomly divided into groups. Each group represents mobile users in the coverage of one SBS. According to the above popularity prediction results, all contents are sorted according to the future requested probability, and the cache order table of each SBS is obtained. Note that the users in the mobile edge network has mobility, the order will be rearranged when the users in the coverage of the SBS changes. There are M BSs and N users in total.

TABLE 1. Proactive caching algorithm.

STARR	
Step 1：	Check whether the request in the backhaul transmission queue has timed out? If "yes"， the timeout request is deleted from the queue and is regarded as a packet loss. If "no"， continue step 2.
Step 2：	When the user requests reach the BS, determine whether the content has been cached at the BS? If "yes", transfer the content to the user directly. If "no", Add the request to the transmission queue and record when the request was generated.
Step 3：	Determine whether the length of the transmission queue is larger than the backhaul bandwidth? If "no"， the MEC server checks its own cached content and proactively request to the remote server to transmit contents with high cache value. If "yes"， the request is processed in the order of the request queue, and the content is transmitted to the base station.
Step 4：	Determine whether the predictive cache of the content is smaller than any content in the BS cache? If "no"， replace the contents with the lowest cache value in the BS cache. If "yes", return to step 2.
Application update	

The backhaul bandwidth is set to be C_b. The user requests for contents come from test set, including F content files. The size of each content is set as L. The transmission rate is set as b. If the content delivery latency is below a certain threshold, then the user's request is satisfied.

On the contrary, if a user request has been queued for longer than the threshold time during peak hours, it is considered lost and no longer transmitted.

D. CONTENT CACHING DELIVERY ALGORITHM BASED ON GREEDY ALGORITHM

After obtaining the predicted content popularity, this paper makes use of greedy algorithm to process the predicted popularity, with the minimum average transmission delay as the optimization goal for content distribution within every server. The greedy algorithm is an algorithm which seems to be the optimal for the time being, so its solution is a local optimal solution in some sense. According to differences of the chosen data type and the greedy model, the performance finally obtained by the greedy algorithm is different.

Define the average transmission delay of the content o_i cached in any server:

$$H_{m,i} = \sum_{\substack{l \in C \\ l \neq m}}^{M} P_{l,i} \cdot d_{l,m} \cdot T_{l,m} \cdot L \tag{6}$$

According to the idea of the Most Popular Content (MPC), the content with the highest popularity and the more concentrated popularity trend should be cached. Therefore, this paper ranks the content popularity according to the content popularity of various servers, and calculates the greedy algorithm of the content whose popularity ranks in a descending order. Based on the calculation results, which content to be delivered to which server is decided. When every content is computed, this paper accounts for the average transmission delay, $H_{m,i}$, which might be brought by the internal storage of every server, chooses the content caching plan with the least transmission expense according to the idea of the greedy algorithm, and judges whether the current server has any caching space unoccupied. If there is, set its $x_{m,i}$ to be 1m, and reduce the caching space of the sever by L. If the server has no caching space unoccupied, then choose the server with the second minimal delay. The following content can be handled according to the above logic until the cache space is fully occupied.

The above greedy algorithm might obtain a local optimal solution; thus, the global optimal solution might not be obtained. For example, in the above algorithm, the content cache proceeds from the upper to the lower in sequence. This means that every content can occupy the position of one unit of length in the practical cooperative cache domain only, and that one popular content will be cached in the server of the cooperative cache domain only. But in fact, the chances are high that a popular content occupies a high degree of popularity in different servers. When the server's caching space is large, the above greedy algorithm might result in redundant caching of some content with a low degree of popularity in some servers. Consequently, the content with a higher popularity in the server is left not cached, and only the local optimum is obtained. In the practical sense, the minimum average transmission delay is not obtained. Instead, the caching space is wasted for the caching of the content that is not so popular.

Concerning the limitation of the computing result stated above, this paper improves the optimum solving to obtain the global optimum. According to the improved algorithm, every server is independently computed with the predicted content popularity as the basis and find out the files with a high popularity but not cached and the files with a low popularity but cached. The two kinds of files are compared to judge whether they can substitute each other.

Define $A = \{A_1, A_2, A_3, \ldots, A_S\}$, which indicates the caching of all data in the cooperative cache domain. Where, A_i denotes the collection of all servers which have cached the content o_i.

$$G_m(i, l) = \Delta H_m(i) - \Delta h_m(l) \tag{7}$$

where,

$$\Delta H_m(i) = \sum_{\substack{s=1 \\ s \notin A_i}}^{M} P_{s,i} \cdot d_{s,\mathrm{n}} \cdot L \cdot T_{s,n} - \sum_{\substack{z=1 \\ z \notin A_i \\ z \notin m}}^{M} P_{z,i} \cdot d_{z,n} \cdot L \cdot T_{z,n} \tag{8}$$

$$\Delta h(l) = \sum_{s=1}^{M} P_{s,i} \cdot L \cdot T_{s,0} - \sum_{\substack{z=1 \\ z \neq A_l}}^{M} P_{z,i} \cdot L \cdot T_{z,n} \cdot d_{z,n} \tag{9}$$

In (8), $\Delta H_m(i)$ denotes the altered delay value in the average transmission delay of the content o_i after the files with a high popularity on the server c_m are replaced. In (9), $\Delta h_m(l)$ denotes the altered delay in the average transmission delay

of the content o_l after the files with a low popularity on the server c_m are replaced. When $G_m(i, l) > 0$, the replacement can effectively improve the global optimization of the average transmission delay. Every server is repeatedly optimized in this way. It is not until the revenue function $G_m(i, l) \leq 0$ that the optimization comes to an end and the final cooperative caching content delivery plan is obtained.

V. SIMULATION AND RESULT ANALYSIS

In this section, the performance of the content caching delivery algorithm – Edge Cooperative Caching – based on the small base station's cooperative caching is assessed. In order to verify the effectiveness of the performance optimization degree, this paper chooses three algorithms introduced by Literature [16], including "Least Frequently Used" (LFU), "Least Recently Used" (LRU) and "Random Replacement" (RR), to compare with Edge Cooperative Caching (ECC) and verify the performance of ECC from the perspective of three major indexes, namely the content hit rate, average transmission delay and content cache space utilization, respectively.

A. PARAMETER SETTINGS OF COOPERATIVE CACHING SYSTEM

In the simulation experiment, the parameters of the simulation experiment are defined as below according to the content data requirements in the cooperative cache domain of the cooperative caching system. Assume that there are four randomly-distributed small base stations (BS) in the cooperative cache domain, which are jointly responsible for all user access in the domain, and one thousand pieces of content with different degrees of popularity in line with *Zipf* distribution are issued in the region. The skewness coefficient of the *Zipf* distribution is set to be 0.6, and represents the concentration of their popularity. The value of very content is normalized to be L (L=1), and the unit time, Δt, of the static content popularity is set to be 1h. In other words, the popularity remains the same within every unit time, Δt. At the same time, the time interval is divided by every second of one hour. The user access obeys the Poisson distribution of the parameter, $\lambda = 5$ (indicating that there are five users visiting the cooperative cache domain visiting every second on average). In the simulation process of LFU, LRU and RR, every server conducts independent algorithm computing and content caching according to their user access. When there is user access, the requested content can be obtained from other cooperative cache servers visited by users. Thus, a cooperative caching system is constituted. In the simulation experiment, this paper mainly compares the strengths and weaknesses of three performance indexes of various algorithms under the condition that various algorithms change in the server caching space of the cooperative cache domain.

B. SIMULATION EXPERIMENTAL RESULTS OF VARIOUS ALGORITHMS AND RESULT ANALYSIS

In the cooperative caching model, when the server's caching space changes, three performance indexes, including the content cache hit rate (HR), average transmission delay (ADL) and content cache utilization (CSU), of the edge cooperative caching algorithm are measured.

As shown in Fig. 4, as the content caching space enlarges, the cache hit rate of the four algorithms increases to different degrees. Due to randomness of RR, and the irrelevance of RR algorithm characteristics to popularity, the content cache hit rate of RR increases slower than that of other algorithms. It is apt to say that RR achieves performance at the sacrifice of the caching space. As the caching space changes, the content cache hit rate of LRU is improved by around three folds as the caching space changes. Comparatively, the content cache hit rate of LFU and ECC is improved by around 170% and 120%, respectively. Comparatively, the content cache hit rate improvement of LRU is more significant. As to the reason behind, LRU can easily cause cache pollution because of the emergent or periodical access. Comparatively, the counter caching of LFU can better resolve the cache pollution resulted from accidental access. ECC, compared with the remaining algorithms, is steadier. When the server's cache space reaches 1,000, the content cache hit rate is around 70%, which improves by 12% and 28% compared with LFU and LRU, respectively, and by nearly one-fold compared with RR.

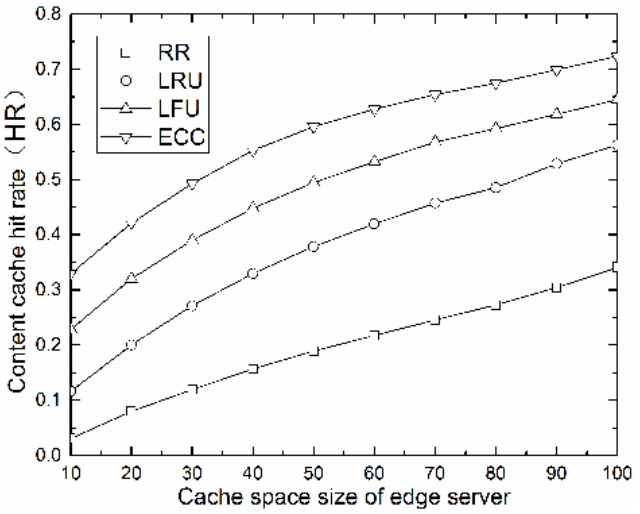

FIGURE 4. Performance comparison of content cache hit rate of different algorithms.

As one observes in Fig. 5, the final average transmission delay of the four cache algorithms all decreases as the content cache space increases. This is because, as the cache space of the cooperative cache server increases, the cooperative cache server can have more popular content cached therein. In this way, when the user is visiting the popular content, there is more popular content that can be directly acquired from the local cooperative cache domain, which can significantly reduce the transmission delay brought about by user access. The average transmission delay performance and the content cache hit rate performance of the four algorithms are the same. Compared with other algorithms, the reduction range

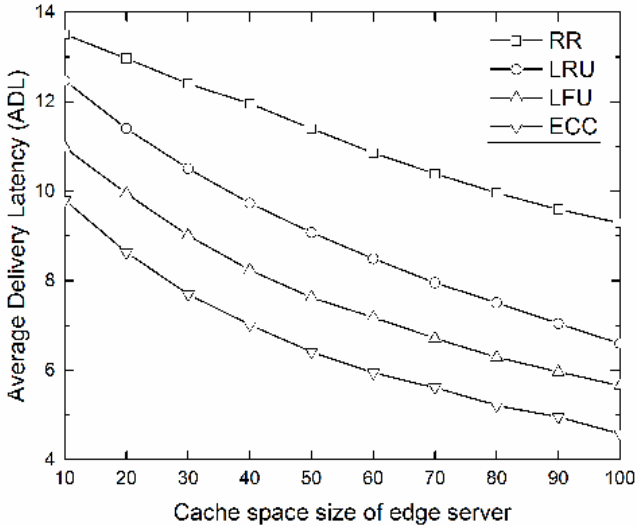

FIGURE 5. Comparison of average transmission delay performance of different algorithms.

of RR's average transmission delay is smaller than that of the rest, thus suggesting a poorer average transmission delay performance. Likewise, the average transmission delay performance of LFU is also superior to that of LRU. As to ECC, its performance in reducing the average transmission delay is outstanding, whose average transmission delay variation is 53% along with the changes of the cache space, which is superior to that of LRU and LFU by 47% and 48%, respectively. To sum up, in terms of average transmission delay performance, ECC outperforms LFU by 19%, LRU by 30% and RR by 51%. Thus, it can be seen that ECC based on the base station cooperative cache outperforms a series of commonly-seen cache elimination algorithms in pursuing the minimum of the average transmission delay.

Fig. 6 shows the strengths and weaknesses of the four algorithms in terms of the content cache space utilization.

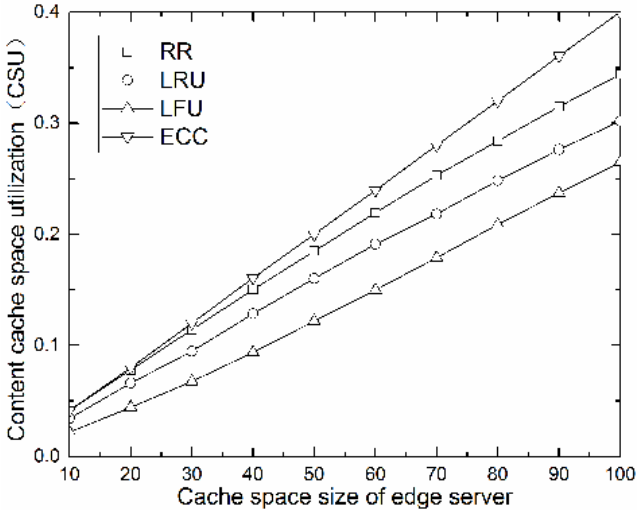

FIGURE 6. Performance comparison of content cache space utilization of different algorithms.

It can be seen that the performance of RR is different from that above, whose content cache space utilization is just behind that of ECC. This is because RR is free from the influence of popularity, which caches and stores the content randomly. Under the condition of excessive concentration of popularity, RR can perhaps perform favorably when the cache space is certain. At the same time, Fig. 6 is found with an interesting phenomenon, that is, the quantity of valid content cached by LFU in the cache space is around 12% smaller than the quantity of effective content cached by LRU in the cache space. The content cache hit rate and the average transmission delay of LFU and LRU might easily lead to the conclusion that LFU outperforms LRU in terms of content cache space utilization, but this is not the actual simulation outcome. According to the comparison of various commonly-seen algorithms in Literature [17], LFU can well predict the popular content during user access to content with a concentrated popularity. As to LRU, some periodical or accident content access can easily cause cache pollution, waste of the cache space and inaccurate grasp of the popular content. The cause of the situation in Fig. 6 is because the concentrated cache capacity of the highly popular files is superior to that of LRU in various servers. Under the assumption that the 1,000 pieces of content with a relatively concentrated popularity and in line with the *Zipf* distribution are distributed, the factors influencing the average content transmission delay of a cooperative cache server domain are not limited to how much content is cached in the cooperative server but also include whether every server has cached the content with a high concentration of popularity [18]–[20]. In light of the assumption made above, chances are high that a content with a relatively high popularity in an integrated cooperative domain might enjoy a high popularity. Under the condition, the user access transmission delay is usually decided by files with a high degree of popularity. Therefore, this paper allows reduction of the average transmission delay through improvement of the repeated cache rate of the same popular content. This can help explain the abnormal performance of LFU and LRU in content cache utilization, and can further substantiate that, after the initial content caching matrix is obtained using the greedy algorithm, the content cache replacement algorithm should also be employed to further optimize the average content transmission from the global perspective.

VI. CONCLUSION

The rapid growth of network traffic and data types in the big data era has posed a tremendous challenge to the traditional network transmission model. Edge cache emerges as a favorable solution to the above problem. The cooperative operation between the edge cache server and the cloud data center can create a preferential user experience, effectively improve the service quality, reduce the redundant network traffic and optimize the network performance. This paper proposes the edge cooperative caching (ECC) algorithm based on the small base station cooperative caching. Experimental results suggest that the cooperative use of the popularity prediction algorithm,

greedy algorithm, and content cache replacement algorithm based on the neural collaborative filtering can help ECC outperform the other three algorithms in that the former can improve the content cache hit rate, reduce the average content transmission delay, and accommodate to the content cache space utilization and effective cache of popular content by various servers.

ACKNOWLEDGMENT

The authors would like to thank the associate professor Z. Lu who is working at Beijing University of Posts and Telecommunications (P.R China) in particularly for the support in thesis examination and guidance. The authors are also grateful to the teachers of OAI WORKSHOP. (http://www.opensource5g.org/) for their invaluable support in deploying the system and in providing experimental data and technical documentation.

REFERENCES

[1] D. Wu, H. Shi, H. Wang, R. Wang, and H. Fang, "A feature-based learning system for Internet of Things applications," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1928–1937, Apr. 2019.
[2] H. Zhang, Y. Dong, J. Cheng, M. J. Hossain, and V. C. M. Leung, "Fronthauling for 5G LTE-U ultra dense cloud small cell networks," *IEEE Wireless Commun.*, vol. 23, no. 6, pp. 48–53, Dec. 2016.
[3] M. Yan, C. A. Chan, W. Li, L. Lei, A. F. Gygax, and C.-L. I, "Assessing the energy consumption of proactive mobile edge caching in wireless networks," *IEEE Access*, vol. 7, pp. 104394–104404, 2019.
[4] H. Zhang, H. Liu, J. Cheng, and V. C. M. Leung, "Downlink energy efficiency of power allocation and wireless backhaul bandwidth allocation in heterogeneous small cell networks," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1705–1716, Apr. 2018.
[5] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.
[6] I. Abdullahi and S. Arif, "Cache-skip approach for information-centric network," *J. Eng. Appl. Sci.*, vol. 11, no. 5, pp. 3413–3418, Mar. 2016.
[7] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Dec. 2012, pp. 1107–1115.
[8] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive content download and user demand shaping for data networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1917–1930, Dec. 2015.
[9] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of distributed caching in D2D wireless networks," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Sevilla, Sep. 2013, pp. 1–5.
[10] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: Moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, Sep. 2016.
[11] X. He, L. Liao, H. Zhang, L. Nie, and X. Hu, "Neural collaborative filtering," presented at the 26th Int. Conf. World Wide Web, Perth, NSW, Australia, Apr. 2017.
[12] M. Zhu, J. Li, N. Wang, and X. Gao, "A deep collaborative framework for face photo-sketch synthesis," in *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 10, pp. 3096–3108, Oct. 2019.
[13] Z. Yang, J. He, and S. He, "A collaborative filtering method based on forgetting theory and neural item embedding," in *Proc. IEEE 8th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, Chongqing, China, May 2019, pp. 1606–1610.
[14] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.
[15] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Singapore, 2017, pp. 1419–1428.
[16] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 38–43, Mar. 2017.
[17] H. Zhang, N. Yang, K. Long, M. Pan, G. K. Karagiannidis, and V. C. M. Leung, "Secure communications in noma system: Subcarrier assignment and power allocation," *IEEE J. Select. Areas Commun.*, vol. 36, no. 7, pp. 1441–1452, Jul. 2018.
[18] D. Wu, Q. Liu, H. Wang, Q. Yang, and R. Wang, "Cache less for more: Exploiting cooperative video caching and delivery in D2D communications," *IEEE Trans. Multimedia*, vol. 21, no. 7, pp. 1788–1798, Jul. 2019.
[19] X. Liu, J. Zhang, X. Zhang, and W. Wang, "Mobility-aware coded probabilistic caching scheme for MEC-enabled small cell networks," *IEEE Access*, vol. 5, pp. 17824–17833, 2017, doi: 10.1109/access.2017.2742555.
[20] D. Wu *et al.*, "Multimedia data transmission mechanism with privacy protection for Internet of Vehicles," *Future Gener. Comput. Syst.*, vol. 99, pp. 609–623, May 2019.

YU CHEN received the Ph.D. degree in communication engineering from the Beijing University of Posts and Telecommunications. He is currently an Associate Professor with Beijing Polytechnic University. His current research interests focus on radio resource and mobility management, software-defined wireless networks, and broadband multimedia transmission technology.

YONG LIU received the bachelor's degree in communication engineering from the Beijing University of Chemical Technology, Beijing, China, in 2004. His research interests include green multimedia communication, QoE-centric networks, and application management.

JINGYA ZHAO received the master's degree from the Beijing Institute of Technology, in 2007. She joined the School of Telecommunications Engineering, Beijing Polytechnic University, in 2000. Her research interests include open wireless networks, QoE management in wireless networks, software-defined wireless networks, and cross-layer design for mobile video applications.

QINGHUA ZHU received the bachelor's degree in computer software and application specialty from Beijing University, Beijing, China, in 2001. He is currently working with Beijing Polytechnic University. His research interests include the area of multimedia communication, QoE management, green energy efficient networks, and network virtualization (SDN).