TPA based content popularity prediction for caching and routing in edge-cloud cooperative network

Bo Yi[1], Fuliang Li[1,3], Yuchao Zhang[2], Xingwei Wang[1,*]
[1]*College of Computer Science and Engineering, Northeastern University, Shenyang, China*
[2]*School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China*
[3]*Beijing National Research Center for Information Science and Technology, Beijing, China*
yibo@cse.neu.edu.cn, lifuliang207@126.com, yczhang@bupt.edu.cn, wangxw@mail.neu.edu.cn

***Abstract*—The rapid development and application of 5G/B5G generate tremendous amount of traffic which in turn cause great burden for the corresponding transmission network. One typical way to address such challenge is to sink the content (e.g., 4K and 8K videos) from the remote cloud to the edge servers. In this case, how to efficiently visiting and getting these contents becomes a new problem, in which the cooperation between cloud and edge should be taken into consideration. In this regard, this work builds an edge and cloud cooperative routing and caching system which consists of three main modules of content popularity prediction, cooperative caching and cooperative routing. Specifically, the content prediction is designed by jointly leveraging the technologies of Long Short-Term Memory (LSTM) and Temporal Pattern Attention (TPA) to dig the traffic features and predict the future content popularity. Based on the prediction results and the technology of reinforce learning, the cooperative caching module designs both a reactive content replacement and an active content caching strategies. After that, the cooperative routing is carried out to help customers visiting and obtaining these content efficiently with the objective of minimizing the overhead. The experimental results indicate that the proposed methods outperform the state-of-the-art benchmarks in terms of the caching hit rate, the average throughput, the successful content delivery rate and the average routing overhead.**

***Index Terms*—Edge-cloud cooperation, Temporal pattern attention, routing, content popularity, caching**

I. Introduction

The rapid development of the fifth generation mobile technology 5G promotes a lot of new technologies and applications such as artificial intelligence, big data, cloud computing, virtual reality and augmented reality, etc. This is because 5G enables the Internet of everything [1], such that all the equipments and devices used to be disconnected can now communicate and cooperate with each other to complete the same task. Besides, such cooperation can happen not only among mobile devices, but also between clouds and edge servers. In this way, the efficiency of data transmission and content retrieving can be greatly improved.

This work is supported by the National Key Research and Development Program of China under Grant No. 2019YFB1802800; the National Natural Science Foundation of China under Grant No. 62002055, 62032013, 61872073; China Postdoctoral Science Foundation under Grant No. 2020M680972; the Postdoctoral Research Fund of Northeastern University of China under Grant No. 20200103; the Fundamental Research Funds for the Central Universities of China under Grant No. N2016012; the Open Research Fund Program of Beijing National Research Center for Information Science and Technology.

Despite this, the opportunities are always accompanied by the challenges. According to the recent statistics from the International Data Corporation (IDC), there will be over 50 billion mobile devices connecting to the Internet [2]. Under this case, two obvious challenges can be easily discovered. On one hand, such massive mobile devices wil generate tremendous traffic data which are far beyond the capacity of clouds. On the other hand, there are also a lot of invalid data which need to be not only cleaned, but also identified [3]. Apart from these easy-to-be-discovered issues, another significant one behind them is that the burden of the corresponding transmission network between cloud and edge increases greatly and proportional with the scale of mobile devices. Such situation further causes many difficulties for obtaining the required contents in remote clouds. For example, 1 billion customers demand the same 8K video content at the same time would inevitably result in network congestion.

The traditional content delivery network paradigm cannot satisfy the requirements of many 5G applications nowadays. Hence, many researches try to integrate the content delivery network with the edge servers, during which the contents are sunk from cloud to edge for the purpose of releasing the burden of clouds [4]. In addition, this also helps releasing the burden of the transmission network, since customers can visit and obtain the contents from closer edge servers. Another technology used to sink contents from cloud to edge is the concept of Mobile Edge Computing (MEC) which offers better Quality of Service (QoS) to customers. Nevertheless, reviewing the state-of-the-art related work (e.g., [5]-[15]), most of them simply remove the problems from cloud to the edge. One extreme condition is that we can cache all the contents in edge servers, which may result in the fact that only 20% (or even lower) of them can be used effectively. Hence, in order to maximize the resource and the content utilization, a balance and cooperation between the cloud and edge should be reached.

Taking the above analysis into consideration, we propose an edge-cloud cooperative system, in which efficient caching and routing methods are designed to achieve high performance content delivery. The caching part is used to decide which contents should be placed in the corresponding regions, while the routing part is used to efficiently connect customers and the expected contents. Then, the main contributions of this work are summarized as follows:

- We propose an edge-cloud cooperative system for high performance content delivery, in which two kinds of cooperative ways are designed. The first way is vertical cooperation, since it defines the cooperative policy between clouds (i.e., centralized controller) and edge servers. The second way is horizontal cooperation, which defines the cooperative policy among edge servers.
- In addition, a TPA and LSTM based content popularity prediction method is proposed. According to the predicted results, we can inactively replace the unpopular contents with high popularity contents to increase the content hit rate.
- Based on the horizontal cooperative model, we design an active caching method which regards each edge server as an intelligent agent. Then, these agents can learn from each other on the basis of the reinforce learning model, such that the proposed caching method can dynamically adapt to customer requests.
- Based on the vertical cooperative model, the global view of the centralized cloud (i.e., controller) is used to calculate the routing path between customers and the expected contents with minimal overhead and latency.

The rest of this work are organized as follows. Section II reviews the related and state-of-the-art work. Section III comprehensively introduces the proposed system framework and problem model. Section IV shows the detailed design of the proposed methods. Section V gives the experimental results and Section VI concludes this work.

II. Related Work

Efficient content delivery is important especially in the MEC environment, since the rapid increasing mobile traffic would cause a large amount of unnecessary data transmission and lower content hit rate which then decrease the QoS of content delivery. To address such issue, the caching and routing are two typical and vital perspectives for content delivery, from which many researches leveraging the technologies of cloud computing, edge computing and AI are proposed.

As explained, the popular contents are moving from cloud to the edge. Hence, many researches (e.g., [5]-[9]) focus on addressing the content delivery problem at the edge. For example, reference [5] tried to address the content delivery problem at the base station. However, the common way is to cache the contents in the edge server instead of the base station. Following this common trend, reference [6] proposed an intelligent content delivery method in wireless networks by deploying intelligent routers at the edge. Such intelligent edge router was used to store information such as content popularity, user mobility and social relationship, etc. Besides, reference [7] defined the concept of MEC servers at the edge. Then, it proposed a MEC intelligent cache analysis platform with multiple algorithms integrated to decide which media files need to be stored in advance to meet the future content demands. Similarly, reference [8] also targeted on the MEC servers. The differences between them are that the former aimed at improving the content caching efficiency, while the latter aimed at minimizing the capital expenditure of service providers. Different from the above work relying on the centralized control, reference [9] proposed a distributed content delivery framework and it enabled each edge server with the ability to accept/deliver requests or forwarding contents. Then, these edge servers could work together to achieve the same goal and maximize the profits earned from the corresponding content delivery.

The optimal content caching is one important aspect for content delivery. In this regard, reference [10] proposed to search the most suitable content to cache at the edge by using the migration learning model. However, such migration model based caching method relied heavily on the context information (e.g., history requirements and social relationship). In this way, the quality of the context used for training determined the performance of this method, while high quality data was always hard to obtain. Similarly, reference [11] adopted another learning model to estimate the popularity of contents dynamically, which was then used to guide the process of caching. Reference [12] studied the dynamic content placement problem on edge servers. In particular, the time-varying characteristics of contents were considered in this work to propose a new caching method which replaced the unpopular contents with popular ones, such that the cache hit rate could be improved. Despite these work used some kinds of learning models, they were actually not intelligent enough. Then, most researches were begin to use the technologies of deep learning and reinforce learning models.

Reference [13] proposed a content caching and routing method based on the reinforce learning model. On one hand, the history content demands were collected as the training data. On the other hand, the edge servers could also learn from each other. Jointly taking the two points into consideration, this work could reduce the abundant traffic and improve the content deliver efficiency. Reference [14] also leveraged the technology of reinforce learning. The difference was that the latter enabled edge servers to decide caching independently according to their own capacities. Then, the remote centralized controller would evaluate the performance and feedback the results to these edge servers, such that these edge servers could further optimize their actions towards better caching and routing. Generally, the content popularity was unknown and reference [15] intended to address the content delivery problem in such situation with the objective of minimizing the average content transmission delay. In this way, this work first proposed a multi-agent framework to decide the content caching based on deep actor-critic reinforcement learning and then it proposed an improved shortest path routing method to reduce the average delay between customers and contents.

Although the above research have achieved good progress, most of them still ignore the importance of cooperation between edge servers and clouds, which may easily fall into the situation of local optimum. Therefore, this work propose two kinds of cooperative models which are the horizontal cooperation and the vertical cooperation. Based on such two

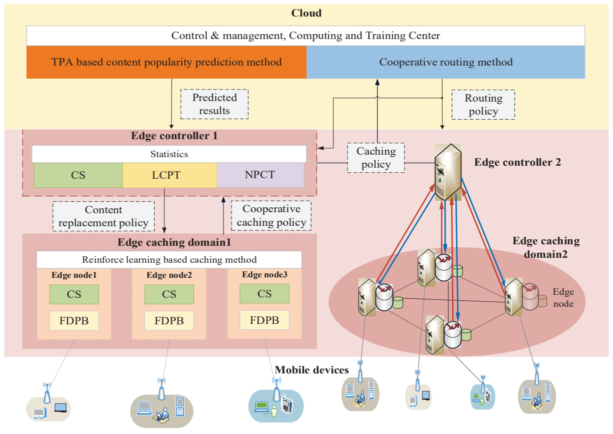


Fig. 1. System framework

cooperation models, we also design the intelligent content popularity prediction, caching and routing methods which can help to increase the utilization of cache hit rate and reduce the transmission overhead.

III. System Framework and Problem Model

A. System Framework

The overall system framework is shown in Fig. 1, where it consists of four main modules, that is, the Statistics Module (SM), the content Popularity prediction Module (PM), the caching module (CM) and the Routing Module (RM). The main functions of them are explained as follows:

- SM: it is implemented in the edge controller and responsible for collecting information from edge servers in its domain with the tables of Content Store (CS), Local Content Popularity Table (LCPT) and Neighborhood Popular Content Table (NPCT). Then, the collected data will be handled in this module before sending them to the cloud for the following content popularity prediction and routing calculation.
- PM: it first constructs a prediction model by jointly leveraging TPA and LSTM. Then, taking the collected and cleaned data from SM as the input, PM will train this prediction model continuously. During the training process, the corresponding prediction parameters will be adapted dynamically to provide the most accurate prediction results for determining the content replacement policy.
- CM: it regards the edge servers as intelligent agents and then build a multi-intelligent-agent reinforce learning model among these agents. Each server is allowed to learn from each other, so that they need to maintain two tables which are the CS and the Forwarding Data Packet Database (FDPD). Then, these edge servers can independently decide which content should be cached according to the actual local conditions. Such pattern is referred to as the horizontal cooperation.
- RM: it is in charge of dynamically planning the routing path between customers and contents under the guidance of the centralized cloud, through which we can build the cooperation between clouds and the edge servers. Such pattern is referred to as the vertical cooperation.

Based on the four main modules, the overall workflow can be described as follows. Firstly, SM collects the related information from each edge server and executes the statistic processing. Then, SM sends the processed data to both the cloud and other edge controllers. On one hand, for the cloud, it regards these data as parameters of both PM and RM. Specifically, PM uses the statistical data uploaded by each edge controller to construct and train the prediction model for the purpose of periodically predicting the content popularity. RM uses these data to calculate the optimal routing path for efficient content delivery. On the other hand, for the other edge servers, they rely on using these data reach a consensus, based on which these edge servers can easily cooperate with each other to find the most balancing and efficient caching policy.

B. Problem Model

In the edge-cloud cooperative environment, there are M edge servers denoted by $S = \{s_1, s_2, \ldots, s_M\}$. Assuming that the centralized cloud has all the contents denoted by $C = \{c_1, c_2, \ldots, c_F\}$, where F is the number of categories of contents. Then, $\forall c_i, s_m$, a binary variable describing the relationship between them is defined as follows:

$$x_{m,i} = \begin{cases} 1,\text{The content } c_i \text{ is cached in } s_m. \\ 0,\ \text{Otherwise.} \end{cases} \tag{1}$$

When one customer demanding the content c_i with the size of l_i, there exist three cases: 1) the content is found in a local server in this domain; 2) the content is found from a server in other domains; 3) the content is found in cloud. Despite this, the overhead per hop per unit size can be unified into η. In addition, if $\exists s_i$ satisfied that $x_{m,i} = 1$, we can calculate the shortest path between the customer and the server s_i easily. Denoting the corresponding hop by hop, then, the overall routing cost can be calculated as follows:

$$\begin{aligned} Minimize: \quad & cost = \eta \times hop \times l_i \\ s.t. \quad & \sum_{i=1} x_{m,i} \times l_i \le Q_m, \forall s_m \in S \\ & x_{m,i} \in \{0,1\}, \forall i \in [1, F] \\ & hop = \min\{hop_{m,i} | x_{m,i} = 1\} \end{aligned} \tag{2}$$

where the first constraint indicates that the size of content cannot exceed the storage capacity of this node; the second constraint means that one content is either cached or not in one node; while the last constraint means that the hop between c_i and s_m should be minimum.

IV. The Proposed Caching and Routing Algorithm

As explained, this work is composed of four main parts which are presented in the following.

A. Data Preprocessing

The content related information are collected by edge controllers to analyze the distribution of contents. Besides, we also extract eight data features to train the prediction model, which are the time, content name, content size, data rate, throughput, delay, request node ID and content popularity. However, the ranges of different data features are different, such that we use the min-max standard to process them in the first place. Given any data feature θ with the range of $[\theta_{min}, \theta_{max}]$, then we have

$$\theta = \begin{cases} \dfrac{\theta - \theta_{min}}{\theta_{max} - \theta_{min}}, \theta_{min} < \theta < \theta_{max}. \\ \quad 1, \quad \theta = \theta_{max} \\ \quad 0, \quad \theta = \theta_{min} \end{cases} \tag{3}$$

Then, these normalized data will be regarded as the input of the prediction model, while the output of this model is the popularity level within the scope of [1,10]. The higher the popularity level, the more popular the content.

B. Content Popularity Prediction

The formats of input and output of the content popularity prediction module are both defined in the above subsection. Hence, in this subsection, we mainly describe the prediction model based on LSTM and TPA. Denoting the hidden state matrix of LSTM by $\mathbf{h} = \{\mathbf{h}_{t-w}, \ldots, \mathbf{h}_{t-1}\}_{(w,m)}$, where w is the size of sliding window and m is the number of variables. The size of convolution kernel is $[w,1]$ and the number of convolution kernel (denoted by C) is k, so that the convolution process is as follows:

$$\mathbf{H}_{i,j} = \sum_{l=1}^{w} \mathbf{h}_{i,(t-w-1+l)} \times C_{j,l}. \tag{4}$$

Now, we introduce the TPA mechanism and calculate the attention score as follows:

$$f(s_t, h_t, c_t) = \mathbf{H}_i \mathbf{W}[h_t; c_t], \tag{5}$$

where $\mathbf{H}_i$ is the hidden matrix; $\mathbf{W}$ is the training parameters; h_t is the hidden state output; c_t is the memory output, at the time t.

Considering the situation that there may be multiple features affecting the output at time t, the Sigmoid active function is applied to obtain the attention score, that is, $\sigma_i = sigmoid(f(s_t, h_t, c_t))$. After getting the score, $\mathbf{H}$ should be weighted to get the context vector v_t, that is, $v_t = \sum_{i=1}^{m} \sigma_i^t \mathbf{H}_i$. Then, according to the above definition, a new hidden state output h_t' can be re-constructed as follows:

$$h_t' = \mathbf{W}_h h_t + \mathbf{W}_v v_t. \tag{6}$$

Then, combing the above definition, we can get the prediction model as follows:

$$y_{t-1+\Delta} = W_{h'} h_t' \tag{7}$$

C. Cooperative Caching

As explained, the content can be hit in three cases. Hence, we assume that the number of content hit in the local domain, the other domains and cloud are denoted by ϕ_0, ϕ_1, ϕ_2 respectively. Since we try to improve the QoS by sinking contents from cloud to the edge, the number of content hit in the local domain should be increased. Then, the objective of cooperative caching is formulated as follows:

$$\begin{aligned} maximize: &\quad \phi_0 - (\phi_1 + \phi_2) \\ s.t. &\quad (2) \end{aligned} \tag{8}$$

The cooperative caching method regards each edge server as an intelligent agent and applies the reinforce learning model to enable each edge server the ability to learn from each other. Hence, we next design the corresponding action space and reward function.

1) Action and State Space Design: According to the basis principle of reinforce learning, the intelligent agents act based on the environment variables. In this work, we regard the remaining cache capacity and the number of content hit as the environment variables. Given any server node s_m, the corresponding state space for such cooperative learning is defined as follows:

$$state\_space_m = \{RemCap_m, Hit_m | m \in [1, M]\}, \tag{9}$$

where $RemCap_m$ is the remaining cache capacity and Hit_m is the number of content hit on the node s_m.

Similarly, given the content c_i and the server node s_m, the action space is designed to be a discrete space, as follows:

$$action\_space_m = \{x_{m,i}^t \in \{0,1\} | s_m \in S, c_i \in C\}, \tag{10}$$

where $x_{m,i}^t = 1$ means that s_m should cache c_i at the time t.

2) Reward Function Design: Based on the environment variables, the intelligent agents will take an action, that is, deciding whether to cache the content or not. After this action, the environment will also feedback a timely reward signal to the intelligent agents. This reward ie either positive or negative, which reflects the affection that taking such action would have on the environment. As for the intelligent agents (i.e., edge servers), they need to maximize the positive reward, such that we define it as follows:

$$reward = -(\alpha hop_{\phi_0} + \beta hop_{\phi_1} + \gamma hop_{\phi_2}), \tag{11}$$

where $hop_{\phi_0}, hop_{\phi_1}, hop_{\phi_2}$ are the hops that the content is hit in the local domain, the other domains and the cloud; α, β, γ are the corresponding weighting parameters.

3) Active Caching and Passive Replacement: Leveraging the above definitions of $state\_space$ and $action\_space$, we establish the reinforce learning model. Training this model until a stable condition is reached, after which we get the content caching matrix $\mathbf{X}^t$ at time t:

$$\mathbf{X}^t = \begin{bmatrix} x_{1,1}^t & x_{1,2}^t & \cdots & x_{1,F}^t \\ x_{2,1}^t & x_{2,2}^t & \cdots & x_{2,F}^t \\ \vdots & \vdots & \ddots & \vdots \\ x_{M,1}^t & x_{M,2}^t & \cdots & x_{M,F}^t \end{bmatrix}. \tag{12}$$

According to the value of any $x_{m,i}^t$, we can actively determine whether to cache the content c_i on the node s_m. Nevertheless, the remaining cache capacity of any node is limited. Hence, once it cannot satisfy the demands, the passive content replacement will be triggered, that is, the content with the lowest popularity will be deleted to release more cache capacity and then the one with high popularity can be cached.

D. Cooperative Routing

The cooperative routing is implemented between the cloud and the edge servers. On one hand, the routing calculation is carried out by the cloud, which relies on using the information provided by the edge servers. On the other hand, the calculated routing policy will be distributed to edge servers from clouds. The overall routing calculating process can be separated into three steps:

- Selecting the content provider node: according to the received interest packet name, the edge domain controller first retrieves the CS and NPCT tables to find server nodes that cache the content, which are defined as the preferred node. Then, calculating the cost from any preferred node to the request node and selecting the one with the lowest cost as the content provider node.
- Calculating the routing path: regarding the content request node as the source and the content provider node as the destination, we use the shortest path to calculate the routing path between them, which is then delivered to edge controllers and used for data forwarding.
- Optimizing the routing process: calculating the routing cost according to (2) and regarding it as another factor to select the corresponding forwarding and routing path.

Generally, the edge nodes will first forward the request data according to its FDPD determined by routing policies from cloud. If no matching item is found in FDPD, this request will be forwarded to the edge controller which then searches the CS and NPCT tables. If there are still no matching items, the edge controller will forward this request to the cloud. As explained, the centralized cloud has the global network view and is able to calculate the optimal routing path. After that, the routing policies will be distributed from cloud to edge controllers and edge servers. However, it is noted that if there is any matching item during the searching process, the request data will be forwarded accordingly.

V. Performance Evaluation

A. Setup

The experimental environment is built using the software of PyCharm toolkit and Keras library. Firstly, for the content popularity prediction part, the prediction step is set to 1 unit and the epoch is set to 50. Secondly, for the caching part, the epoch is set to 1000 and the weighting parameters of the reward function are set to $\alpha = 1, \beta = 5, \gamma = 10$ respectively. Thirdly, for the network routing part, the Uni-C topology is adopted, which has 15 nodes and 17 links distributed in three domains. The experimental hardware is Intel(R) Core(TM) i5-8250U CPU@1.60GHz and Intel(R) UHD GPU Graphics 620.

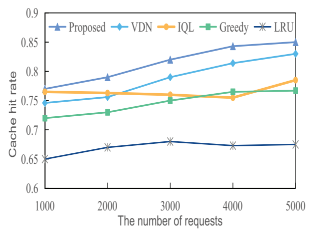
TABLE I
Content popularity prediction results

Prediction step	**k=1**		
Prediction model	***MAE***	***RMSE***	***R^2***
SimpleRNN	0.029	0.036	0.950
LSTM	0.024	0.037	0.964
Att-LSTM	0.027	0.024	0.979
Bi-LSTM	0.021	0.018	0.972
TPA-LSTM	0.011	0.012	0.981

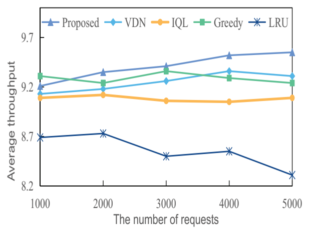
B. Results

1) Content popularity prediction results: In order to show the benefits of the proposed content popularity prediction model (i.e., TPA-LSTM), we calculate the Mean Absolute Error (MAE), the Root Mean Squard Error (RMSE) and the determination coefficient R^2 for TPA-LSTM and the other four benchmarks (i.e., SimpleRNN, LSTM, Att-LSTM and Bi-LSTM). The corresponding results are summarized in Table 1 under the setting that $k = 1$. Specifically, on one hand, we can easily see that the MAE and RMSE: 1) of TPA-LSTM are 0.011 and 0.012; 2) of Bi-LSTM are 0.021 and 0.018; 3) of Att-LSTM are 0.027 and 0.024; 4) of LSTM are 0.024 and 0.037; 5) of SimpleRNN are 0.029 and 0.036. Via comparison, TPA-LSTM has the smallest MAE and RMSE. Generally, the smaller the values of MAE and RMSE, the better the prediction results. Hence, we can conclude that the proposed TPA-LSTM has the best performance, while the overall prediction performance of SimpleRNN is the worst. On the other hand, for the metric of R^2, the higher the better, since it is proportional to the fitting effect. Similarly, we can observe that TPA-LSTM has the highest R^2, while that of SimpleRNN is the smallest. That is because TPA-LSTM can extract features from the hidden-state vectors of LSTM, which enables to select data across multiple time steps and thus to learn efficiently.

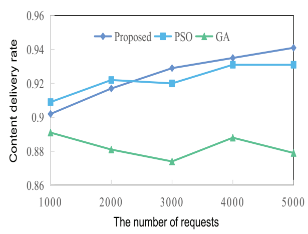
2) Content caching and routing results: The proposed method is firstly compared with the caching benchmarks of Value Decomposition Network (VDN), Independent Q-Learning (IQL), Greedy and Least Recently Used (LRU) over the metrics of the cache hit rate and the average throughput. The corresponding results are shown in Fig. 2(a) and (b), where 2(a) corresponds to the cache hit rate and 2(b) corresponds to the average throughput. In particular, the proposed method has the highest cache hit rate, followed by VDN, IQL, Greedy and LRU. Note that cache hit rate of the proposed method increases along with the increasing of the number of requests, while that of IQL decreases in the first place and then increases. That is because the proposed method has a global caching view from the cloud. Then, it can not only passively carry out the content replacement policy, but also actively cache the most popular contents accordingly. In addition, the proposed method supports the vertical cooperation which is a lack in other benchmarks, such that the proposed method can execute the intelligent content popularity prediction before caching. Then, the prediction results provide better guidance



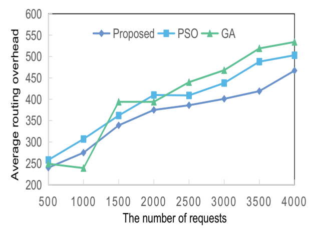
(a) Cache hit rate



(b) Average throughput



(c) Content delivery rate



(d) Average routing overhead

Fig. 2. Experimental results

for caching determination. As for the average throughput, it is not absolutely proportional to the number of requests. For example, the throughput of LRU shows a decreasing trend along with the increasing of the number of requests, because LRU is not able to process such large amount of requests. On the contrary, the proposed method perform well when the number of request increases and it achieves the highest throughput in most cases. One different point should be noticed is that the average throughput of the proposed method is lower than that of Greedy when the number of request is 1000.

Then, we compare the proposed method with the routing benchmarks of Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) over the metrics of the content delivery rate and the average routing overhead. The corresponding results are shown in Fig. 2(c) and (d), where 2(c) corresponds to the content delivery rate and 2(d) corresponds to the average routing overhead. For the content delivery rate, we can see that the performance of the proposed method improves when the number of request becomes large, while that of the other two benchmarks fluctuate greatly. Besides, the average content delivery rate of the proposed method is about 1% 4% higher than the other two methods. That is because the PSO and GA both work on edge controllers, while the proposed method support the vertical cooperation and can work on the basis of a global network view. Then, it can make better decisions instead of falling into the local optimum situation. As for the routing overhead, we can see that the proposed method has the lowest overhead, followed by PSO and GA. It is aware that with the vertical cooperation model, we can quickly locate the expected content source, which directly reduces a lot of unnecessary searching process, that is, the routing hops can be reduced and then the overhead is naturally reduced. However, for PSO and GA, they need to plan a lot of redundant paths before finding the most optimal one, which is actually costly.

VI. Conclusion

By establishing the vertical cooperation model between cloud and edge server as well as the horizontal cooperation model among edge servers, we propose a content popularity prediction based caching and routing method for high efficient content delivery. In particular, the content popularity prediction model is based on TPA and LSTM to achieve high prediction accuracy, while the caching is implemented based on the reinforce learning model to form a feedback closed loop. The routing is implemented to build the optimal connection between customers and contents via the two cooperative models. Despite this, future work are still required to improve the adaptability and flexibility of the proposed method.

References

[1] C. Zhang, Y. Ueng, C. Studer and A. Burg, Artificial Intelligence for 5G and Beyond 5G: Implementations, Algorithms, and Optimizations, IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2020(10)(2): 149-163.

[2] International Data Corporation, Smartphone Growth to Reach Its Highest Level, 2021.05, Available online: www.idc.com.

[3] D. Jiang, L. Huo and H. Song, Rethinking Behaviors and Activities of Base Stations in Mobile Cellular Networks Based on Big Data Analysis, IEEE Transactions on Network Science and Engineering, 2020(7)(1): 80-90.

[4] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan and X. Chen, Convergence of Edge Computing and Deep Learning: A Comprehensive Survey, IEEE Communications Surveys & Tutorials, 2020(22)(2): 869-904.

[5] K. Zhang, S. Leng, Y. He, et al., Cooperative content caching in 5G networks with mobile edge computing, IEEE wireless communications, 2018(25)(3): 80-87.

[6] T. Hou, G. Feng, S. Qin, and W. Jiang, Proactive content caching by exploiting transfer learning for mobile edge computing, IEEE Global Communications Conference, Singapore, 2018: 1-6.

[7] P. Yang, N. Zhang, S. Zhang, et al., Content popularity prediction towards location-aware mobile edge caching, IEEE transactions on multimedia, 2018(21)(4): 915-929.

[8] D. Lee, J. Choi, J.H. Kim, et al., On the existence of a spectrum of policies that subsumes the least recently used and least frequently used policies, SIGMETRICS, 1999(99): 1-4.

[9] X. Zhou, M. Zhao, M. Wu, An in-network caching scheme based on betweenness and content popularity prediction in content centric networking, IEEE international symposium on personal, indoor, and mobile radio communications, Valencia, Spain, 2016: 1-6.

[10] E. Batug, M. Bennis, and M. Debbah, A transfer learning approach for cache-enabled wireless networks, International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Mumbai, India, 2015: 161-166.

[11] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang, Learningbased content caching and sharing for wireless networks, IEEE Transactions on Communications, 2017(65)(10): 4309-4324.

[12] H. Song, S. Chae, W. Shin, et al., Predictive Caching via Learning Temporal Distribution of Content Requests, IEEE Communications Letters, 2019(23)(12): 2335-2339.

[13] W. Jiang, G. Feng, S. Qin, et al., Multi-Agent Reinforcement Learning Based Cooperative Content Caching for Mobile Edge Networks, IEEE Access, 2019(7): 61856-61867.

[14] Y. Zhang, B. Feng, W. Quan, et al., Cooperative Edge Caching: A Multi-Agent Deep Learning Based Approach, IEEE Access, 2020(8): 133212-133224.

[15] C. Zhong, M. Gursoy, S. Velipasalar, Deep Multi-Agent Reinforcement Learning Based Cooperative Edge Caching in Wireless Networks, IEEE International Conference on Communications (ICC), 2019: 1-6.