Cooperative Edge Caching Based on Temporal Convolutional Networks

Xu Zhang, Zhengnan Qi, Geyong Min, Wang Miao, Qilin Fan, and Zhan Ma

Abstract—With the rapid growth of networked multimedia services in the Internet, wireless network traffic has increased dramatically. However, the current mainstream content caching schemes do not take into account the cooperation of different edge servers, resulting in deteriorated system performance. In this paper, we propose a learning-based edge caching scheme to enable mutual cooperation among different edge servers with limited caching resources, thus effectively reducing the content delivery latency. Specifically, we formulate the cooperative content caching problem as an optimization problem, which is proven to be NP-hard. To solve this problem, we design a new learning-based cooperative caching strategy (LECS) that encompasses three key components. Firstly, a temporal convolutional network driven content popularity prediction model is developed to estimate the content popularity with high accuracy. Secondly, with the predicted content popularity, the concept of content caching value (CCV) is introduced to weigh the value of a content cached on a given edge server. Thirdly, an novel dynamic programming algorithm is developed to maximize the overall CCV. Extensive simulation results have demonstrated the superiority of our approach. Compared with the state-of-the-art caching schemes, LECS can improve the cache hit rate by 8.3%-10.1%, and reduce the average content delivery delay by 9.1%-15.1%.

Index Terms—Cooperative edge caching, temporal convolutional networks, content caching value, content popularity

1 Introduction

The rapid development of 5G/B5G technology and the prevalence of smartphones have catalysed the proliferation of emerging multimedia services, such as Cloud Virtual Reality (CloudVR) and Cloud Gaming. In addition, the global mobile data traffic will increase by more than six times from 2017 to 2022, with an annual growth rate of 42% [1]. As a sequence, tremendous pressure has been brought to the current wireless communication system in terms of the network latency and throughput. In this context, Multi-access Edge Computing (MEC) has been introduced to cope with the growing data traffic and strict latency requirements via bringing computing and storage closer to the network edge [2], [3], [4], [5].

Content caching is one of the key pillars of MEC, which can reduce the redundant transmission of data content, network latency and bandwidth consumption, as well as improve the Quality-of-Experience (QoE) perceived by users. In this regard, a rich literature has been developed around the design of content caching mechanisms [6], [7], [8]. However, most of the conventional caching approaches follow fixed rules, and thus cannot cope with the frequently changing content popularity and object access patterns. Recently, learning-based approaches [9], [10], [11], [12] have been proposed. For example, Li *et al.* [9] proposed a PopCaching system to increase the cache hit rate, which uses a learning-based method to predict content popularity. In [10], the authors proposed a Seq2Seq model to predict the content popularity. However, these studies on estimating the content popularity suffer from either slow training speed, gradient disappearance or insufficient accuracy. Moreover, none of the above methods considered the cooperation between servers. When the local server does not cache the requested content while the neighboring servers cache the content, the non-cooperative caching approaches will forward the request to the remote central server, resulting in repeated data transmission and resource consumption.

There also exist some research works [13], [14], [15] in terms of cooperative content caching. For instance, Serbetci *et al.* [14] designed an optimal caching strategy by estimating the cost function, and proposed a transfer learning based method to estimate the content popularity to improve the cache hit rate. Chen *et al.* [15] proposed a cooperative caching algorithm based on collaborative filtering, and

- *Xu Zhang, Geyong Min, and Wang Miao are with the Department of Computer Science, College of Engineering, Mathematics, and Physical Sciences, University of Exeter, EX4 4QF Exeter, U.K. E-mail: {x.zhang6, G.Min, wang.miao}@exeter.ac.uk.*
- *Zhengnan Qi and Zhan Ma are with the School of Electronic Science and Engineering, Nanjing University, Nanjing, Jiangsu 210093, China. E-mail: Zhengnan@smail.nju.edu.cn, mazhan@nju.edu.cn.*
- *Qilin Fan is with the School of Big Data and Software Engineering, Chongqing University, Chongqing 400030, China, and also with the Chongqing Key Laboratory of Digital Cinema Art Theory and Technology, Chongqing University, Chongqing 400030, China. E-mail: fanqilin@cqu.edu.cn.*

Manuscript received 30 Mar. 2021; revised 24 Nov. 2021; accepted 25 Nov. 2021. Date of publication 14 Dec. 2021; date of current version 31 Jan. 2022.
This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB2100804, in part by the EU Horizon 2020 Research and Innovation Programme under the Marie Sklodowska-Curie under Grant 898588, in part by the EU Horizon 2020 INITIATE Project under Grant 101008297, in part by the National Natural Science Foundation of China under Grants 61902178, 92067206, 62102053, and 61972222, in part by the Natural Science Foundation of Jiangsu under Grant BK20190295, in part by the Leading Technology of Jiangsu Basic Research Plan under Grant BK20192003, and in part by the Chongqing Key Laboratory of Digital Cinema Art Theory and Technology under Grant 2021KF01.
(Corresponding authors: Geyong Min and Wang Miao.)
Recommended for acceptance by J. Zhai.
Digital Object Identifier no. 10.1109/TPDS.2021.3135257

adopted a greedy algorithm to obtain the approximate minimum total content delivery latency. However, the greedy algorithm falls into local optimal solutions in some cases [15]. In addition, most of the existing works assume that the contents have the same size, which does not hold in practical systems. For example, the size of a popular video is much larger than the size of a picture with a high click-through rate. How to obtain the global near-optimal solution with the minimum total content delivery latency faces high challenges.

In this paper, we propose a learning-based edge caching scheme to enable mutual cooperation among different edge servers with limited caching resources, in order to maximize the overall caching performance. The cooperative content caching problem is formulated as an optimization problem with the objective of minimizing the total latency of content delivery, which is proved to be NP-hard. To this end, we design a learning-based edge cooperative caching scheme (LECS), which possesses the following three key components. Firstly, we propose a temporal convolutional network (TCN) driven content popularity prediction model (TCNCP), which has the advantages of high parallelism, fast convergence and stable gradient. To the best of our knowledge, this is the first of its kind to leverage TCN to predict the future popularity of contents. Next, based on the content popularity, we comprehensively consider several other factors (i.e., content delivery delay, and content size) that affect the caching performance, and define a concept for the content caching value (CCV) to weigh the value of a content cached on a given edge serve. The CCV is calculated based on the content delivery latency, content popularity and size. Finally, we design a dynamic programming optimization scheme based on the CCV. The extensive performance comparison with with the state-of-the-art approaches reveals that LECS can improve the content cache hit rate by 8.3%-10.1% and reduce the average content delivery latency by 9.1%-15.1%.

The main contributions of this paper can be summarized as follows:

- Firstly, we propose a new cooperative caching strategy named LECS to enable mutual cooperation among different edge servers with limited caching resources, with the aim of maximizing the overall caching performance.
- Secondly, we develop a temporal convolutional network driven content popularity prediction model (TCNCP) to predict the future content popularity, which can strike a good balance between long and short-term memory and obtain accurate prediction results.
- Thirdly, we introduce the concept of content caching value (CCV) to weigh the value of a content for a given edge server. Compared with the content popularity, CCV takes into account more pragmatic factors such as content delivery delay and content size.
- Next, we propose a dynamic programming driven caching strategy to maximize the overall CCV. Theoretical analysis and results demonstrate that the strategy can achieve near-optimal content placement.
- Last but not least, the results obtained from real-trace driven simulation experiments show that LECS can achieve superior performance in terms of the average content delivery latency and cache hit rate in different scenarios.

The rest of the paper is organized as follows. Section 2 presents a review of related work on caching approaches. Section 3 introduces the system model and the formulation of the cooperative content caching problem. In Section 4, we present in detail our approach to predicting content popularity and present the TCNCP model. Furthermore, we elaborate on the CCV and LECS. In Section 5, we conduct the performance evaluation by comparing our algorithm with the state-of-the-art approaches. Finally, Section 6 concludes the paper.

2 Related Work

There exist extensive studies on the design of caching mechanisms, which can be divided into the following two categories.

2.1 The Non-Cooperative Caching Approaches

Caching approaches were originally derived from page replacement algorithms in the operating system or storage system [16], such as First Input First Output (FIFO), Least Recently Used (LRU) and Least Frequently Used (LFU) [17], [18]. These classical approaches are easy to be implemented, paving the way for their wide adoption in Content Delivery Network (CDN) [19]. However, these approaches with fixed rules can hardly adapt to dynamic requirements of users, thus deteriorating the performance under dynamic scenarios. To fill the gap, Psara *et al.* [20] proposed a probability-based caching strategy for data resources, which gives high priority to content streams over the longer paths and allocates less space than shorter paths, thus improving the overall resource utilization. In [21], a node centrality driven heuristic method was proposed to minimize the content delivery latency via a shortest path tree algorithm. A joint caching and routing strategy [22] was designed in consideration of the capacity constraints of small base stations. Poularakis *et al.* [23] proposed to optimize the caching policies based on multicast transmissions via a random rounding technology, which can yield significant energy savings. Gu *et al.* [24] proposed a greedy strategy to solve the cache space allocation problem of macro base stations, which can obtain a cache space allocation result close to the optimal solution. Traverso *et al.* [25] proposed a Shot Noise Model which captures the dynamic content popularity, effectively resolving the temporal locality of content popularity. Gharaibeh *et al.* [26] proposed an online caching algorithm that allows users to access multiple base stations, minimizing the total cost of content providers. Zhong *et al.*[27] presented a DRL-based framework with Wolpertinger architecture for content caching at the base station, which improves cache hit rate. Sugi *et al.* [28] proposed the T-Caching which operates in a distributed way using tokens to maximize cache hit rate. However, most of the existing work assumes that the user requests are evenly distributed [29] without considering the cooperation between MEC servers. When applying these non-cooperative algorithms into real-world scenarios, it is easy to cause the adjacent MEC servers to cache duplicate contents, resulting in cache redundancy and system inefficiency.

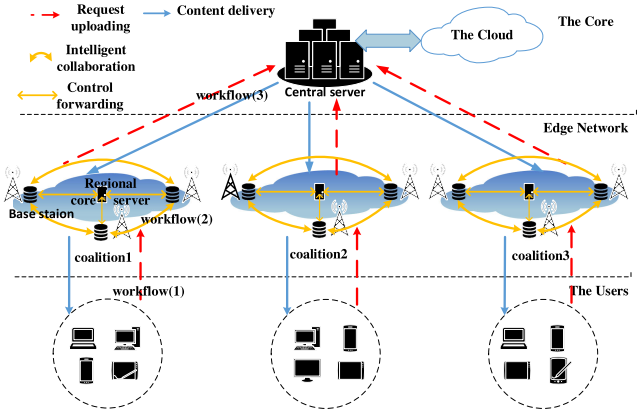


Fig. 1. The structure of cooperative caching system.

2.2 The Cooperative Caching Approaches

To cope with the aforementioned limitations, cooperative caching approaches have attracted an increasing interest from both academia and industry [30]. Serbetci *et al.* [14] designed an optimal random caching strategy by estimating the cost function, and proposed a method based on transfer learning to estimate the content popularity, with the aim of improving the cache hit rate. Li *et al.* [31] proposed two caching strategies that mine user/group interests to improve caching performance at network edge, using recommendation algorithms to predict the probability of contents in the next time slot. Zhao *et al.* [32] considered a content caching structure that combines distributed caching and centralized processing to reduce redundant backhaul traffic and improve the quality of service (QoS). Sun *et al.* [33] presented a caching scheme assisted by simulated annealing algorithm, which optimizes bandwidth allocation through double decomposition to ensure user fairness with a low interruption probability. Wang *et al.* [34] proposed a zone-based cooperative content caching and distribution scheme and developed a heuristic cooperative content caching strategy, which divides the storage space in each MEC server into two parts. The first part caches locally popular contents and the second part is used to cooperatively cache zone-wide popular items, with the assumption that the content popularity is known as a prior. Nie *et al.* [35] proposed a Bayes-based learning algorithm that learns the popularity profile, then optimizes the content placement by using greedy algorithm to cache contents with better channel qualities. An edge cooperative cache based on neural collaborative filtering and greedy algorithm was presented in [15] to accurately predict content popularity and minimize content transmission latency. Xia *et al.* [36] proposed an online algorithm named CEDC-O based on Lyapunov optimization to minimize the cost. However, most studies assume that all contents have the same size, while the sizes of contents are diversified in practice.

3 System Model and Problem Formulation

3.1 System Overview

In this work, we take the position that an effective MEC system should embed cooperative caching mechanisms in its architecture, as shown in Fig. 1. It consists of the central servers, MEC servers co-located with the base stations, regional core servers, and users. The base stations connected with each other through optical fiber belong to a cooperative coalition, while the central server distributes contents to base stations via backhaul links. Within the same cooperative coalition, an MEC server can share contents with each other to maximize the overall resource utilization. A cooperative cache coalition covers a large cell and provides proxy services. MEC servers in different cooperative coalitions do not exchange information to ensure user privacy. A regional core server connects directly to all the edge servers in the cooperative coalition to gather the information of servers.

When a user requests a content via an MEC server, the overall system workflow is as follows:

- Firstly, the local MEC server checks whether it has cached the content requested by the user. If yes, the local MEC server sends the content to the user directly; otherwise, the local MEC server will go to the second step to find out whether the nearby server has the content.
- Secondly, the regional core server finds out whether other servers in the cooperative coalition have cached the content. If yes, the server which has the content with the least latency delivers the content to the local server and then the local server sends it to the user; otherwise, the local MEC server will go to the third step to get the data resource on the central server.
- Thirdly, once the content is not in the cooperative coalition, it will be transmitted by the central server to the local server and then sent by the local server to the user.

Note that, fetching a content from the central server not only increases the network load but also increases the latency. Therefore, caching the content in the local MEC servers or cooperative coalition can reduce the content delivery latency and data traffic. However, the caching capacity is limited on the MEC servers and the requests from users arrive unevenly distributed. Hence, how to design an efficient caching strategy is challenging.

3.2 Problem Formulation

The terminologies defined in this section is listed as Table 1. For brevity, we focus on one cooperative cache coalition. Assume that there are M MEC servers deployed within a cooperative coalition, where the set of MEC servers is $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$, and the corresponding caching capacity is $\mathcal{C} = \{c_1, c_2, \ldots, c_M\}$. Let s_0 represent the central server and c_0 denote its capacity, which is infinite. Assume that the central server provides F different contents and the corresponding set is $\mathcal{O} = \{o_1, o_2, \ldots, o_F\}$, and the corresponding size of each content is $\mathcal{L} = \{l_1, l_2, \ldots, l_F\}$. Each user independently requests content $o_i (1 \leq i \leq F)$ from an MEC server within the cooperative coalition. We define the popularity of content o_i at time t on the server s_m as $P_{m,i,t}$. We assume that within a given time interval, the content popularity is static. The cache information can be shared between various servers on a cooperative cache coalition. We set a content cache matrix $X = (x_{m,i})_{M \times F}$, where

TABLE 1
The Main Terminologies Used in the Description of LECS

Notation	Semantics
M	Number of the MEC servers
$\mathcal{S}$	Set of the MEC Servers $\{s_1, s_2, \ldots, s_M\}$
$\mathcal{C}$	Caching capacities of the MEC Servers $\{c_1, c_2, \ldots, c_M\}$
$\mathcal{O}$	Set of contents $\{o_1, o_2, \ldots, o_F\}$, where F is the number of contents.
$\mathcal{L}$	Size of the contents $\{l_1, l_2, \ldots, l_F\}$, where F is the number of contents
$P_{m,i,t}$	Popularity of content o_i at time t on the server s_m
$\overline{P}_{T+1,m,i}$	Predicted content popularity of content o_i on s_m at time point $T+1$
$x_{m,i}$	Indicator whether content o_i is cached on the server s_m or not
$d_{s_n,s_m,i}$	Delivery latency of content o_i from server s_n to server s_m
$D_{m,i}$	Server with the minimum content delivery latency when distributing the content o_i to s_m
$y_{m,i}$	Latency for the MEC server s_m to obtain content o_i
X_T	Set of historical popularity $\{P_{m,1,T}, P_{m,2,T}, \ldots, P_{m,f,T}\}$ of all contents on MEC server s_m at time T
$V_t(s_m, o_i)$	CCV of the content o_i on the MEC server s_m
β	Skewness coefficients of a Zipf distribution

$$x_{m,i} = \begin{cases} 1 & \text{If content } o_i \text{ is cached on the server } s_m, \\ 0 & \text{Otherwise.} \end{cases} \tag{1}$$

The caching scheme is updated for each time interval. Without loss of generality, we consider the scheme design in the time interval $[t, t+\Delta t)$. During the time interval, the pattern of user's demand on content does not change. Based on the prediction of the content requirement pattern within $[t, t+\Delta t)$, a cooperative caching deployment strategy will be designed to maximize the QoE perceived by users. Considering that the latency is one of the most critical issues that affects the QoE, we choose the content delivery latency as the index to evaluate our system. The latency perceived by a user can be divided into three components, including the content transmission latency from the central server to an MEC server, the content sharing latency between MECs, and the content fetching latency from a local MEC to a user. In summary, the edge caching problem is equivalent to minimizing the overall latencies perceived by users with the constraints of caching capacities.

The delivery latency for a unit size of data from server s_n to server s_m consists of two parts: 1) the transmission latency $d^{tl}_{s_n,s_m}$ and 2) the propagation latency $d^{pl}_{s_n,s_m}$, where the transmission latency is the time elapsed from the first bit until the last bit of a content has left the transmitting node, while the propagation latency refers to the time needed for the head of the signal to travel from the sender to the receiver. Hence, the delivery latency $d_{s_n,s_m,i}$ for content o_i from server s_n to server s_m is as follows:

$$d_{s_n,s_m,i} = d^{tl}_{s_n,s_m} \cdot l_i + d^{pl}_{s_n,s_m}. \tag{2}$$

The server with the minimum content delivery latency when distributing the content o_i to s_m is expressed as follows:

$$D_{m,i} = \arg\min_{s_n} \{d_{s_n,s_m,i} | \forall s_n \in \{S \cup \{s_0\}\} \& x_{n,i} = 1\}. \tag{3}$$

The content delivery latency $y_{m,i}$ for the MEC server s_m to obtain content o_i is $d_{D_{m,i},s_m,i}$, where the content o_i is forwarded from server $D_{m,i}$ to server s_m. Specifically,

- When the content o_i is cached on the MEC server s_m, $D_{m,i} = s_m$, and $y_{m,i} = 0$.
- When the content o_i is not cached on s_m but exists on another server in the cooperative coalition, the server with the least delivery latency will be selected to outsource the content. In such a case, $D_{m,i} \in S$, and $y_{m,i} = d_{D_{m,i},s_m,i}$.
- If the content o_i is not cached in any MEC server in the coalition, it will be transferred from the central server to the local MEC and distributed to the user. In such a case, $D_{m,i} = s_0$, and $y_{m,i} = d_{s_0,s_m,i}$.

Therefore, the cooperative caching can be formulated as an optimization problem with the objective to minimize the average content delivery latency while conforming to the caching capacities of MEC servers. Specifically,

Optimization objective:

$$(P1) \min \sum_{m=1}^{M} \sum_{i=1}^{F} P_{m,i,t} \cdot y_{m,i} \tag{4}$$

Constraints:

$$s.t. \sum_{i=1}^{F} x_{m,i} \cdot l_i \leq c_m, \forall s_n \in S, \tag{5}$$

$$x_{m,i} \in \{0,1\}, \forall s_n \in S, \forall o_i \in O, \tag{6}$$

$$x_{0,i} = 1, \forall o_i \in O. \tag{7}$$

Objective (5) is to minimize the content delivery latency, where $P_{m,i,t}$ represents the popularity of content o_i in server s_m at time t. Constraint (6) ensures that the total sizes of all the cached contents at server s_m should not exceed the server's capacity. Constraint (7) represents that the cloud data center stores all the contents.

Lemma 1. *P1 is an NP-hard problem.*

Proof. Consider a special use case for the optimization problem P1: suppose that there is only one MEC server s_m in the cooperative coalition. To minimize the average latency perceived by users fetching contents from s_m, s_m should cache the contents to achieve the lowest average content delivery latency and make full use of the cache

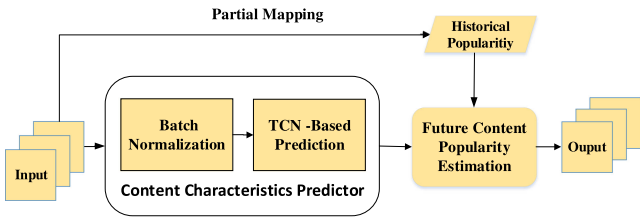


Fig. 2. The TCNCP model.

space. The content in this problem only has the option to be cached or not, where no content can be partially cached. In this scenario, the original problem P1 is transformed into a knapsack problem, which is known as a classical NP-hard problem. Since the subproblem of the original optimization problem P1 is NP-hard, P1 is an NP-hard problem. □

4 Learning-Based Cooperative Caching Mechanism

We propose a learning-based cooperative caching mechanism named LECS to solve the above problem. The LECS strategy can be divided in three steps. First, a learning-based model is designed to predict the content popularity in the future. Then, the content caching value (CCV) is introduced to value the content on a specific MEC server. Finally, based on the CCV, a dynamic programming based algorithm is proposed to make the decisions for addressing the cooperative caching problem.

4.1 Content Popularity Prediction Based on Temporal Convolutional Network (TCNCP)

Content popularity is an important parameter in the design of edge caching strategies. Most of the previous work assume that the content popularity follows the Zipf distribution [10], [15], [37], [38], [39]. Actually, due to the rapid update of content and dynamic changes in user demand, content popularity is difficult to be obtained in advance, and can only be estimated based on the relevant information in the past. In this paper, we exploits the TCNCP to predict the future content popularity.

The TCNCP Model Components: As shown in Fig. 2, the model consists of two components: the content characteristics predictor and the future content popularity estimation. When the characteristics of content is captured by the predictor component, it will be forwarded to the estimation component. Then, the estimation component takes advantage of the popularity data of several time slots in the past, and leverages the weighted-index average method to weighted sum with it to estimate the future popularity of the content. Therefore, our model can achieve a balance between long-term and short-term burst memory.

Content Characteristics Predictor. It mainly encompasses a batch normalization module and a TCN-based prediction module. The former module aims at improving the stability of the overall neural model via normalization of the inputs, while the latter module is responsible for predicting the popularity characteristics of future contents through analyzing the past content request information.

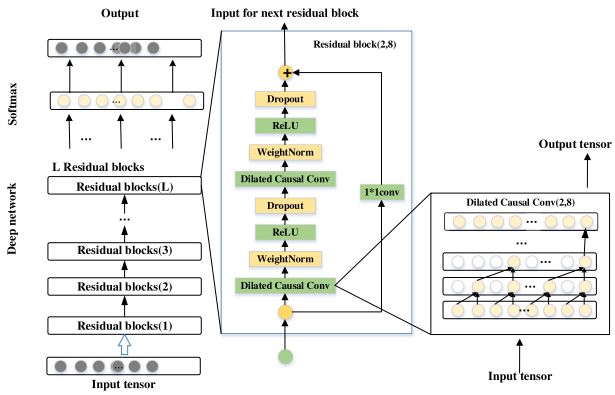


Fig. 3. TCN prediction for caching.

Future Content Popularity Estimation. It estimates the future popularity of contents by balancing the content popularity predicted by the Content Characteristics Predictor with the content popularity in the past.

4.1.1 Content Characteristics Predictor

The objective of this module is to construct the appropriate input sequence and predict the reasonable expected output based on TCN, which can help LECS to make effective cache decisions. Specifically, the content popularity feature vectors from time $(T-k)$ to time T are taken as input, that is, the input to the TCN-based Prection is $\{X_{T-k}, X_{T-k+1}, X_{T-k+2}, \ldots, X_T\}$, where $X_T = \{P_{m,1,T}, P_{m,2,T}, \ldots, P_{m,f,T}\}$ is the set of popularity characteristics of all contents on MEC server s_m at time T, and f is the number of contents. The expected output is a vector $\{Y_{T+1}, Y_{T+2}, \ldots, Y_{T+K}\}$, which represents the collection of the popularity characteristics of all contents in the future K time slots, where $Y_{T+1} = \{\hat{P}_{m,1,T+1}, \hat{P}_{m,2,T+1}, \ldots, \hat{P}_{m,f,T+1}\}$. Our goal is to construct an effective input-output mapping, thus predicting the popularity characteristics of future contents from historical request information.

To achieve this aim, TCN is exploited to realize the mapping [40], [41], [42], [43]. TCN has the following advantages to predict the content popularity. Firstly, the content popularity prediction problem can be transformed into a time series prediction problem, while TCN does a fantastic job of sequence modeling; Secondly, compared with the typical recurrent neural network (RNN) structure that has always been used for sequence modeling, TCN can be processed in parallel at a large scale, so the network speed will be faster during training and testing; Thirdly, TCN can effectively avoid the problem of gradient explosion and disappearance; Last but not least, the training process of TCN takes up less memory, especially for long sequences, which can better extract the features of the time series for prediction. The overall architecture of TCN is shown in Fig. 3.

In detail, TCN adopts a 1-D fully-convolutional network (FCN) architecture [42], where each hidden layer has the same length as the input layer. TCN can achieve intensive prediction with the FCN, which helps to sense the information of the entire input sequence. Besides, the convolution in the TCN architecture has a causal relationship, which means that the future information will not leak into the past in the time series forecast. In what follows, we will elaborate

how the current convolution structure is integrated into TCN by considering both deep network and long-term dependence.

Dilated Casual Convolution. Causal convolution requires a plethora of layers or large convolution kernels to widen the receptive field, which is necessary for the construction of long-term memory. To solve this problem, we introduce dilated convolution into the model. By applying the dilated convolution, the model has a larger size of receptive field. Fig. 3 shows the dilated casual convolution of TCN. For the filter $\mathcal{F} = \{f_1, f_2, \ldots, f_k\}$, the dilated casual convolution y_t of $\mathcal{X} = \{x_1, x_2, \ldots, x_T\}$ at x_t:

$$y_t = \sum_{k-1}^{K} f_k x_{t-(K-k)d}, \tag{8}$$

where d is the dilation factor, and k is the filter size.

Residual Block. The residual block contains two dilated causal convolutions. The weight of convolution kernel is normalized, and a dropout is added to TCN after every dilated causal convolution in the residual block to realize regularization. However, the input directly adds the output vector of the residual function in the standard ResNet [44], and the input and output may have different widths in TCN. Hence, TCN adds 1×1 convolution to ensure that the corresponding pixels between F (X) and X have the same dimension. The entire process is illustrated in Fig. 3.

4.1.2 Future Content Popularity Estimation

To improve the performance of TCN, the short-term sudden memory and long-term memory are balanced in this module, where different priorities of contents at different time points are considered for the the popularity prediction. Specifically, the popularity of contents in the past n time slots and the short-term future popularity generated by the *Content Characteristics Predictor* are considered, and the final popularity data is obtained by weighted summation with the exponential average method. Then, at the next moment of time $T+1$, the estimated content popularity $\overline{P}_{T+1,m,i}$ of content o_i in the s_m area of the server is :

$$\overline{P}_{m,i,T+1} = (1-\lambda)\hat{P}_{m,i,T+1} + \sum_{t=T-n+1}^{T} \lambda^{T-t+1} P_{m,i,t}, \tag{9}$$

where $P_{m,i,t}$ is the popularity of content o_i at the t-th time, $\hat{P}_{m,i,T+1}$ is the popularity of content o_i at time $T+1$ predicted by the *Content Characteristics Predictor*, λ is a constant $(0 < \lambda < 1)$ to adjust the proportion between historical data and latest data, and n is the length of historical data to be considered. This method combines the historical content popularity information to effectively prevent the ever-changing popularity incurred by high user dynamics, thus balancing short-term sudden memory and long-term memory.

4.2 The Content Caching Value

Most of the existing caching approaches based on content popularity is to cache the most popular contents in each MEC server in exchange for the maximum cache performance. However, they fail to take the cooperations between MEC servers into consideration. If an MEC server only selects the content with the high local popularity, multiple MECs in the same coalition may cache the same content ineffectively, resulting in cache redundancy. The system performance may be deteriorated if we consider the popularity only without the factors of content size and delivery latency.

To address this challenge, we introduce a novel metric named CCV. The CCV is calculated from the perspective of a cooperative coalition, which takes into account factors such as the content popularity, content size, and delivery delay. In particularly, when users request the content o_i from the cooperative coalition, the average delivery latency perceived by them when requesting o_i from the coalition is given by:

$$\sum_{n=1}^{M} d_{D_{n,i},s_n,i} \cdot P_{n,i,t}, \tag{10}$$

where $D_{n,i}$ is the MEC server with the minimum content delivery latency to provide content o_i to s_n, and $P_{n,i,t}$ is the popularity of o_i on the server s_n at time point t. Then the CCV $V_t(s_m, o_i)$ which weighs the value of o_i on s_m can be defined in the following two cases.

Case One. If the server s_m does not cache the content o_i, the average latency perceived by users requesting the content o_i from the cooperative coalition after s_m caches the content o_i is as follows:

$$\sum_{n=1}^{M} d_{E_{n,i},s_n,i} \cdot P_{n,i,t}, \tag{11}$$

where $x_{m,i} = 0$, and

$$E_{m,i} = \arg\min_{s_k} \{d_{s_k,s_n,i} | \forall s_k \in \{S \cup \{s_0\}\}, x_{k,i} = 1 \text{ if } k \neq m\}.$$

Then the CCV $V_t(s_m, o_i)$ is defined as the benefit if s_m caches the content o_i:

$$\begin{aligned} V_t(s_m, o_i) &= \sum_{n=1}^{M} d_{D_{n,i},s_n,i} \cdot P_{n,i,t} - \sum_{n=1}^{M} d_{E_{n,i},s_n,i} \cdot P_{n,i,t} \\ &= \sum_{n=1}^{M} \max\{d_{D_{n,i},s_n,i} - d_{s_m,s_n,i}, 0\} \cdot P_{n,i,t}, \end{aligned} \tag{12}$$

Case Two. If the server s_m does cache the content o_i, the average latency perceived by users requesting the content o_i from the cooperative coalition after s_m removes the content o_i is as follows:

$$\sum_{n=1}^{M} d_{F_{n,i},s_n,i} \cdot P_{n,i,t}, \tag{13}$$

where $x_{m,i} = 1$, and

$$F_{m,i} = \arg\min_{s_k} \{d_{s_k,s_n,i} | \forall s_k \in \{S \cup \{s_0\}/\{s_m\}\}, x_{k,i} = 1\}.$$

Then the CCV $V_t(s_m, o_i)$ is defined as the loss if s_m removes the content o_i:

$$V_t(s_m, o_i) = \sum_{n=1}^{M} d_{F_{n,i},s_n,i} \cdot P_{n,i,t} - \sum_{n=1}^{M} d_{D_{n,i},s_n,i} \cdot P_{n,i,t}$$
$$= \sum_{n=1}^{M} \max\{d_{F_{n,i},s_n,i} - d_{s_m,s_n,i}, 0\} \cdot P_{n,i,t} \quad (14)$$

4.3 Dynamic Programming Based Decision Making

Based on the CCV, we transfer the original problem P1 into another optimization problem P2, which aims at maximizing the overall CCV in the coalition. Specifically,

Optimization objective:

$$(P2)\max \sum_{m=1}^{M} \sum_{i=1}^{k} V_t(s_m, o_i) \cdot x_{m,i} \quad (15)$$

Constraints:

$$s.t. \sum_{i=1}^{k} x_{m,i} \cdot l_i \leq c_m, \forall s_m \in S, \quad (16)$$

$$x_{m,i} \in \{0,1\}, \forall s_m \in S, \forall o_i \in \mathcal{O}. \quad (17)$$

where $x_{m,i}$ indicates whether s_m should cache the content o_i. To solve the problem, a Dynamic Programming (DP) based algorithm is put forward, which divides the whole decision-making process into several single-stages and solves them one by one. These stages with multiple states and decision variables can be deduced forward based on a recursive relationship. In this way, the optimal solution to the original problem can be obtained if the starting stage can be solved optimally. Specifically, the stages, states and recursive relationship for P2 are illustrated as follows.

Stages and States. We divide the whole dynamic programming process into k stages, that is, $\{stage[1], \ldots, stage[i], \ldots, stage[k]\}$, where k is the number of newly requested contents plus the contents in the cache. At each stage, let z_i indicate the cache decision for a specific content. We represent the state at $stage[i]$ as $res[i,j]$, where $res[i,j]$ is the maximum value of the cumulative CCV for the first i contents.

$$res[i,j] = max \sum_{k=1}^{i} V_t(s_m, o_k) \cdot w_k. \quad (18)$$

$$s.t. \sum_{k=1}^{i} w_k \cdot l_k + j \leq c_m, \quad (19)$$

where w_k indicates whether content o_k should be cached at the stage i to achieve the maximum value $res[i,j]$ of the cumulative CCV for the first i contents.

Recursive Relationship. In stage $stage[1]$, since there is only one file to be cached, when the remaining cache space is larger than the size of this content, the best decision is $z_{1,1} = 1$. In $stage[i]$, LECS first checks whether the remaining cache space can cache the content o_i, and then determines if caching the content that can get the optimal value and the optimal solution for the $res[i,j]$.

Lemma 2. *The optimal solution at the stage[i] can be obtained according to the following state transfer equation:*

$$res[i,j] = \begin{cases} max\{res[i-1,j], res[i-1,j-l_{id_i}] \\ +V_t(s_m, o_{id_i})\} & l_{id_i} \leq j, \\ res[i-1,j] & l_{id_i} > j. \end{cases} \quad (20)$$

Boundary condition

$$res[i,j] = 0, \text{when } i = 0 \text{ or } j = 0. \quad (21)$$

Proof. We first consider a caching placement for i-th content. When the remaining space $j \geq l_{id_i}$, there are two cases. When s_m stores content o_{id_i}, the remaining caching capacity becomes $j - l_{id_i}$ and $res[i,j]$ becomes $res[i-1, j-l_{id_i}] + V(s_m, o_{id_i})$; otherwise, $res[i,j]$ becomes $res[i-1,j]$. When the remaining space $j \leq l_{id_i}$, $res[i,j]$ becomes $res[i-1,j]$. □

Finally, the optimal decision process Z is obtained from the state transition equation. The dynamic programming algorithm is summarized in Algorithms 1 and 2.

Algorithm 1. Dynamic Programming Cache Placement Algorithm Based on the CCV

Input: Cache space: c_m; $A = [a_0, \ldots, a_{k-1}]$, where k is the total number of newly requested content plus the content in the cache.
Output: Decision process: $Z = [z_0, \ldots, z_{k-1}]$.
// a_i=($o_{id_i}, l_{id_i}, r_{id_i}$);
// o_{id_i}: content item;
// l_{id_i}: the size of content o_{id_i};
// r_{id_i}: the CCV of o_{id_i};

$Z \leftarrow [0, \ldots, 0]$;
$res \leftarrow [[0, \ldots, 0], \ldots, [0, \ldots, 0]]$;
for $i = 1$ to k **do**
  for $j = 1$ to c_m **do**
    if $a[i-1][1] > j$ **then**
      $res[i][j] = res[i-1][j]$;
    else
      $res[i][j] = max\{res[i-1][j], res[i-1][j - a[i-1][1]] + a[i-1][2]\}$;
    end if
  end for
end for
GetCacheList(A, res, Z, k, c_m);
Update Z

Algorithm 2. Function GetCacheList()

function GetCacheListA, res, Z, i, j
  if $i > 0$ **then**
    if $res[i][j] == res[i-1][j]$ **then**
      $z_{i-1} = 0$;
      GetCacheList($A, res, z, i-1, j$);
      else if $j - a[i-1][1] >= 0$ and $res[i-1][j - a[i-1][1]] + a[i-1][2]$ **then**
      $z_{i-1} = 1$;
      GetCacheList($A, res, Z, i-1, j - a[i-1][1]$);
    end if
  end if
end Function

5 Evaluation

In this section, we first compare the performance of the proposed scheme with the benchmark methods in terms of cache hit rate and content delivery latency. Then, we verify the effectiveness of the TCNCP model by comparing it with the classical and state-of-the-art content popularity prediction methods.

5.1 Experiment Setup

5.1.1 Default Parameter Settings

In the simulation experiments, we consider that there are four base stations in a cooperative cache coalition, and each base station is equipped with an MEC server. Without loss of generality, we assume that the content providers publish 1000 video files whose popularity follows the Zipf distribution model [45], and the size of each content is randomly selected from {1,3,5,7,9} [46], [47]. The caching capacity of each server is set as 256, and the skewness coefficient of the Zipf distribution is set to be 0.55. We set the range of $d_{s_0,s_m,i}$ as [10,20] ms, $d_{s_n,s_m,i}$ as [2,4] ms, which are the representative configuration in the 5G era. In order to simplify the experiment, we assume that the delivery latency of $d_{s_n,s_m,i}$ is equal to $d_{s_m,s_n,i}$.

For the TCNNCP prediction model, we leverage Random Search [48] to set the parameters, which samples the search space instead of brute forcing all possible parameter sets, thus avoiding the curse of dimensionality. In detail, the residual network depth is set to 10, the dilations are set as [1, 2, 4, 8], and the kernel size is set as 2. The loss function is chosen as Mean Absolute Error (MAE). We set the length of the sliding window to 2000, the sliding step length to 200, and the length of input to 20 time steps. We aim to predict the content popularity of the next 5 future units, that is, K = 5.

5.1.2 The Benchmark Approaches

To intuitively and effectively illustrate the advantages of the LECS, we compare it with the following benchmark approaches:

- *LFU (Least Frequently Used)* [17]: The cache node calculates the number of arrivals of all contents. When the storage space is full, the least frequently used content is replaced.
- *LRU (Least Recently Used)* [17]: The cache node records all the recently arrived contents. When the storage space is full, the least recently used one is replaced by the new one.
- *ECC (edge cooperative caching)* [15]: The strategy uses the neural collaborative filtering algorithm to predict content popularity, and adopts a greedy algorithm to obtain the cache deployment.
- *Distributed* [32]: The Distributed strategy makes caching decisions from the perspective of the whole cooperative coalition. The strategy ranks the contents according to the popularity of the whole cooperative coalition, selects the content with higher ranking to cache, and only caches one copy of content item in the cooperative coalition. The policy does not cache duplicates, so it can cache as much content as possible.
- *DeepCache* [10]: The DeepCache considers the content popularity of the local server region, and leverages LSTM Encoder-Decoder model to predict the future characteristics of an object, which is used to guide the caching decision of LRU.

5.1.3 The Performance Metrics

To weigh how LECS can improve the system performance, we leverage two metrics including content cache hit rate (HR) and average content delivery latency (ADL).

On the one hand, HR is defined as the percentage of requests whose required content is cached in the cooperative coalition. Specifically,

$$\mathrm{HR} = \frac{N}{R} \tag{22}$$

where R represents the number of requests received by the cooperative coalition per period Δt and N is the number of cache hits.

On the other hand, ADL is defined as the average latency experienced by the requests from the cooperative coalition. Specifically,

$$\mathrm{ADL} = \min \sum_{m=1}^{M} \sum_{i=1}^{F} P_T(s_m, o_i) \cdot y_{m,i} \tag{23}$$

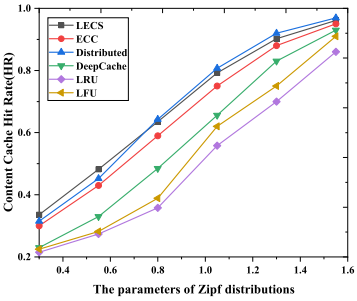
where $P_T(s_m, o_i)$ represents the popularity of content o_i at time T on the server s_m and $y_{m,i}$ is the content delivery latency. The definition of $y_{m,i}$ is given in Section 3.

5.2 Caching Performance

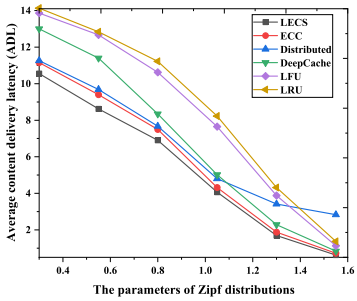
We explore the impact of system parameters on the caching performance, including the size of the caching capacity, the number of service contents, the skew coefficient of the popularity distribution, and the cooperative area.

5.2.1 Impact of Zipf Distribution Parameters

To analyze the impact of content popularity on performance, we adjust the parameter β of the Zipf distribution to change the content popularity distribution. We observe that as the parameter β increases, the caching performance improves. When the parameter β becomes higher, more caching spaces are allocated to contents with high popularity. Fig. 4 shows that as β increases, the performance gap between the various solutions gradually increases. We notice that the performance achieved by *Distributed* strategy is close to that by the *LECS* strategy, and even better in some cases, as shown in Fig. 4a. This is because the *Distributed* strategy is to cache as much contents as possible without duplicates. However, in the case when some contents become highly popular on different servers, the average content delivery latency achieved by the *Distributed* strategy will increase, as shown in Fig. 4b. When β is large, most user requests are concentrated on a small amount of contents, and the *Distributed* strategy does not cache popular content on each server. Compared with the *ECC* strategy, the *LECS* strategy improves the content cache hit rate by 1.1%-12.2% and reduces the average content delivery latency by 5.3%-13%.

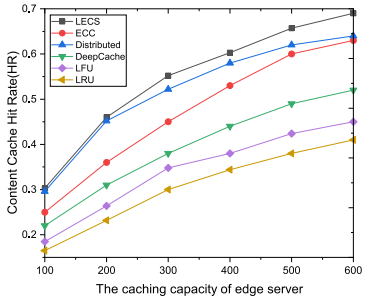


(a)

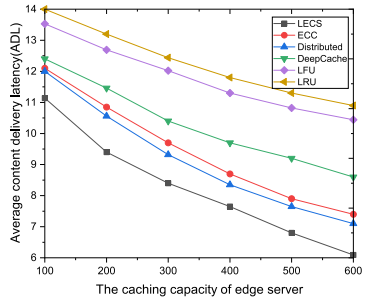


(b)

Fig. 4. The impact of the Zipf parameters on the caching performance: a) HR; b) ADL.



(a)



(b)

Fig. 5. The impact of the caching capacity on the caching performance: a) HR; b) ADL.

5.2.2 *Impact of Caching Capacity*

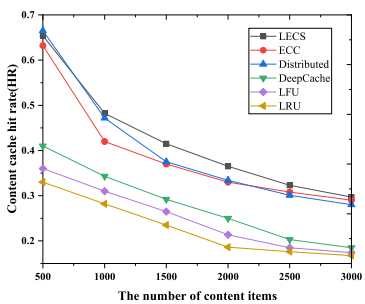
In order to investigate the impact of the caching capacity on the caching performance, we adjust the caching capacity of each MEC server ranging from 100 to 600, while keeping the other parameters as their default values. It is intuitive that the caching performance gets better with the increase of the the caching capacity. This is because that more contents can be cached at the MEC servers, and the requests of users can be served within the coalition rather than from the remote cloud. As depicted in Fig. 5a, the cache hit rate of all cache strategies increases as the caching capacity increases, while the *LECS* strategy outperforms all the other benchmark approaches. Similarly, Fig. 5b shows that the ADL decreases with the increasing of the caching capacities and the *LECS* strategy has the lowest ADL. Compared with *ECC*, the *LECS* strategy can reduce the ADL by 4.3%-10.6%.

5.2.3 *Impact of the Number of Contents*

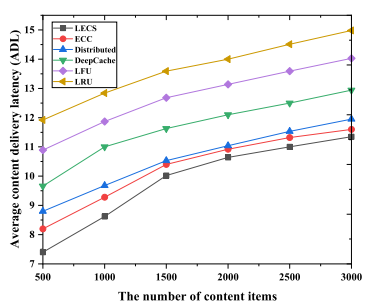
Furthermore, we also evaluate the performance of the *LECS* strategy under different numbers of contents. To this end, we set the number of content items increasing from 500 to 3000 and keep the other system parameters fixed, the HR decreases as the number of contents increases, and the ADL increases as the number of contents increases. When the number of contents increases, more and more requests from the users cannot be served from the local cache with limited cache capacity. As shown in Fig. 6a, the *LECS* mechanism improves the *HR* by 6.1%-12.2% compared to the *ECC* strategy, and is better than the *Distributed* strategy to a certain extent. Moreover, the *LECS* strategy reduces the *ADL* by 4.3%-11% compared to the *ECC* strategy, as illustrated in Fig. 6b.

5.2.4 *Impact of the Cooperative Area*

To investigate the impact of cooperative coalition area on the caching performance, we adjust the number of MEC servers in the cooperative coalition as the number of MEC servers is always positively correlated to the era of the

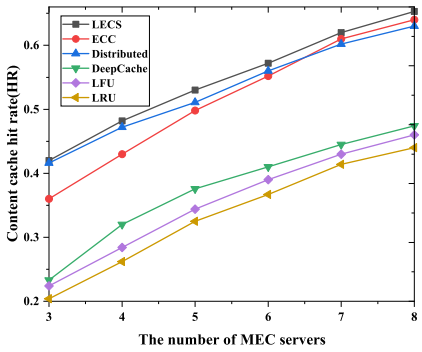


(a)

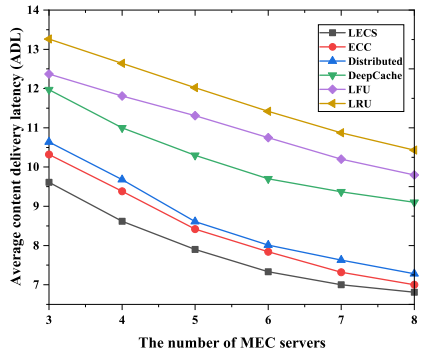


(b)

Fig. 6. The impact of the number of content items on the caching performance: a) HR; b) ADL.



(a)



(b)

Fig. 7. The impact of the number of MEC servers on the caching performance: a) HR; b) ADL.

coalition. In detail, the number of MEC servers is set from 3 to 8. As depicted in Fig. 7, the *HR* under all strategies increases as the number of MEC servers increases, while the *ADL* decreases, which is in accordance with our intuition that contents are much more likely to be cached if there exist more MEC servers. Experimental results show that the *LECS* strategy is better than other baseline cache strategies. Specifically, compared with the second-ranked *ECC* strategy, *LECS* improves the *HR* by 3.6%-12.1% and reduces the *ADL* by 4.6%-7.1%. We notice that the performance of the *DeepCache* strategy is improved slowly as the number of MEC servers increases. As the *DeepCache* strategy only considers the content popularity of local MEC server, it is easy to cache multiple duplicate content and cause redundancy when the number of servers increases.

TABLE 2
Prediction Accuracy

Dataset	Algorithm	MSE	MAE
Dataset1	DeepCache	2.21×10^{-6}	1.135×10^{-3}
	ECC	2.204×10^{-6}	1.133×10^{-3}
	TCNCP	2.087×10^{-6}	1.104×10^{-3}
Dataset2	DeepCache	2.491×10^{-6}	1.159×10^{-3}
	ECC	2.502×10^{-6}	1.160×10^{-3}
	TCNCP	2.045×10^{-6}	1.012×10^{-3}
Dataset3	DeepCache	2.187×10^{-6}	0.975×10^{-3}
	ECC	2.179×10^{-6}	0.970×10^{-3}
	TCNCP	2.075×10^{-6}	0.934×10^{-3}

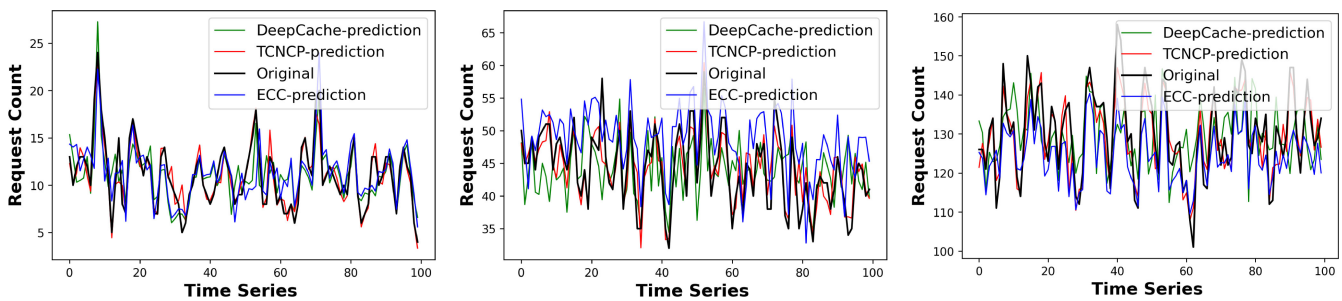
5.3 TCNCP Prediction Accuracy

To verify the effectiveness of the proposed *TCNCP* model when predicting the content popularity, we compare *TCNCP* with the state-of-the-art approaches, *ECC* [15] and *DeepCache* [10] on three different datasets, where the popularity of contents has different distributions. Specifically, the skewness coefficients β of Dataset1, Dataset2 and Dataset3 are set as 0.3, 0.55, and 0.8, respectively. And the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) are selected as the metrics to weigh the prediction accuracy [10]. As illustrated in Table 2, compared with *DeepCache* and *ECC*, *TCNCP* has the lowest error rate in terms of both MSE and MAE, which indicates that TCNCP has superior performance in predicting the content popularity. To visualize the comparison, Fig. 8 shows the original and the predicted content popularity by the three approaches over time. Likewise, *TCNCP* outperforms *DeepCache* and *ECC* in tracking the original time series.

To explore the impact of the hyper-parameters of TCNCP on the performance, we have set the parameters to different values and measured the prediction accuracy of the model. As illustrated in Fig. 9, TCNCP has the highest prediction accuracy when λ is set to 0.2 while n is set to the half of the input length. When keeping the other parameters as their default values, the MAE of the predicted content popularity can be 14% higher when λ is set to 0.2, while 8% higher when n is set to one fifth of the input length.

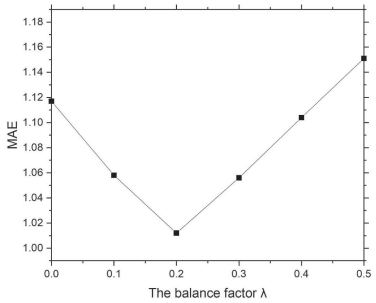
5.4 Superiority of CCV Compared With Content Popularity

Instead of making caching decisions by considering content popularity only, LECS introduces CCV to weigh the value of content on a specific edge server and determines what to

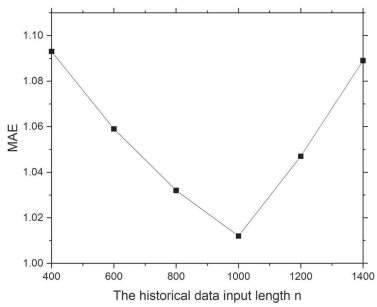


(a) β=0.3 (b) β=0.55 (c) β=0.8

Fig. 8. The impact of Zipf parameter on the performance of the TCNCP model: a) $\beta = 0.3$; b) $\beta = 0.55$; c) $\beta = 0.8$.



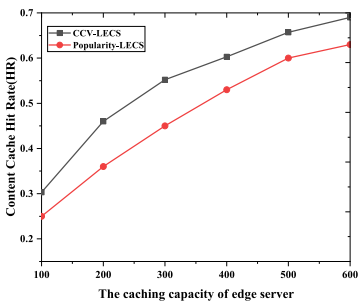
(a)



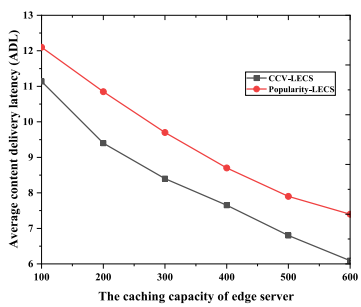
(b)

Fig. 9. Impact of parameters on the prediction accuracy of TCNCP.

cache on which edge server by maximizing the overall CCV. In this subsection, we verify the superiority of CCV by comparing the proposed LECS (to make it clear, we represent it as CCV-LECS in this subsection) with its alternative (represented as Popularity-LECS), which are solely depending on content popularity. Different from CCV-LECS, Popularity-LECS skips the procedure of computing CCV and replaces CCV with the ratio of popularity to content size in the dynamic programming. Fig. 10 shows the changes of the cache hit rate and the average content delivery latency of the two methods as the cache space increases, respectively. As shown in the figure, the CCV-LECS can achieve more superior performance in terms of the two metrics compared with the Popularity-LECS. Specifically, CCV-LECS strategy can improve the content cache hit rate by 9%-20% and reduce the average content delivery latency by 7.9%-15%.

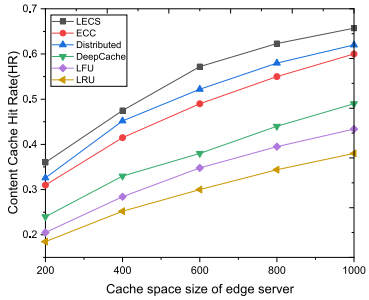


(a)

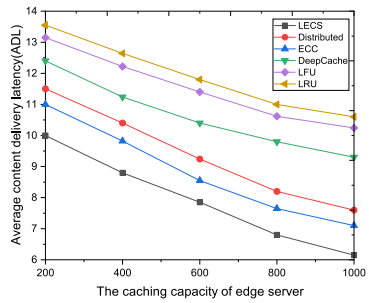


(b)

Fig. 10. The impact of caching capacity on the caching performance: a) HR; b) ADL.



(a)



(b)

Fig. 11. The impact of the number of MEC servers on the caching performance with real-world datasets: a) HR; b) ADL.

5.5 Verification With Real-World Datasets

To verify the performace of LECS when applied in a real system, we evaluate it with a real-world dataset named MovieLens [49]. The dataset was collected by GroupLens Research, which encompasses more than 1 million records related to 3,952 movies and 6,040 users. Specifically, the UserID, the MovieID, the Timestamp and other user information are included in the dataset, which can be utilized to calculate the content popularity of different contents. The other settings of our experiment keep the same as above. We adjust the caching capacity of each MEC server ranging from 200 to 1000. Fig. 11 shows that as the caching capacity increases, the cache performance of all cache strategies increases as the caching capacity increases. Experimental results show that the *LECS* strategy can achieve the best performance among all the strategies. Compared with the *ECC*, the *LECS* can improve the HR by 8.3%-10.1%, and reduce the ADL by 9.1%-15.1%.

6 Conclusion

In this paper, we propose a learning-based cooperative edge caching approach to improve the caching performance. We formulate the cooperative edge caching problem as a

NP-hard knapsack problem with the goal of minimizing the average content delivery latency. To solve the problem, we firstly establish a TCNCP model to predict the popularity of future contents. This model can effectively balance short-term and long-term memory, thus achieving accurate predictions on the content popularity. Then, we define the concept of CCV, which can take various factors into account, such as the content delivery latency and the content size. Finally, based on the CCV, we propose a dynamic programming caching strategy, which can obtain the near-optimal cache placement scheme. In order to evaluate the performance of the proposed LECS strategy, we compare it with five benchmark algorithms from five aspects, including the prediction accuracy, content popularity distribution, caching capacity, number of contents, and cooperative area. Simulation-driven experiments and performance results show that LECS can achieve the best performance in terms of the cache hit rate and the average content delivery latency.

ACKNOWLEDGMENTS

This article reflects only the authors' view. The European Union Commission is not responsible for any use that may be made of the information it contains. Xu Zhang and Zhengnan Qi contributed to this work equally.

REFERENCES

[1] Cisco visual networking index: Forecast and methodology, 2017–2022. Accessed: Feb. 29, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] M. Patel *et al.*, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-Edge Computing Industry Initiative*, vol. 29, pp. 854–864, 2014.

[3] X. Zhang *et al.*, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 178–183, Aug. 2019.

[4] R. Bi, Q. Liu, J. Ren, and G. Tan, "Utility aware offloading for mobile-edge computing," *Tsinghua Sci. Technol.*, vol. 26, no. 2, pp. 239–250, 2021.

[5] L. Liu, X. Chen, Z. Lu, L. Wang, and X. Wen, "Mobile-edge computing framework with data compression for wireless network in energy internet," *Tsinghua Sci. Technol.*, vol. 24, no. 3, pp. 271–280, 2019.

[6] Q. Fan, X. Li, S. Wang, S. Fu, X. Zhang, and Y. Wang, "NA-Caching: An adaptive content management approach based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 152 014–152 022, 2019.

[7] Y. Zhao *et al.*, "Caching salon: From classical to learning-based approaches," in *Proc. IEEE Int. Conf. Serv.-Oriented Syst. Eng.*, 2019, pp. 269–2695.

[8] X. Xu, H. Li, W. Xu, Z. Liu, L. Yao, and F. Dai, "Artificial intelligence for edge service optimization in internet of vehicles: A survey," *Tsinghua Sci. Technol.*, vol. 27, no. 2, pp. 270–287, 2022.

[9] S. Li, J. Xu, M. van der Schaar, and W. Li, "Popularity-driven content caching," in *Proc. IEEE INFOCOM Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[10] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "DeepCache: A deep learning based framework for content caching," in *Proc. Workshop Netw. Meets AI ML*, 2018, pp. 48–53.

[11] S. M'ller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.

[12] T. Trzci'ski and P. Rokita, "Predicting popularity of online videos using support vector regression," *IEEE Trans. Multimedia*, vol. 19, no. 11, pp. 2561–2570, Nov. 2017.

[13] H. Tan, S. H.-C. Jiang, Z. Han, L. Liu, K. Han, and Q. Zhao, "Camul: Online caching on multiple caches with relaying and bypassing," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 244–252.

[14] B. Serbetci and J. Goseling, "Optimal geographical caching in heterogeneous cellular networks with nonhomogeneous helpers," 2017, *arXiv:1710.09626*.

[15] Y. Chen, Y. Liu, J. Zhao, and Q. Zhu, "Mobile edge cache strategy based on neural collaborative filtering," *IEEE Access*, vol. 8, pp. 18475–18482, 2020.

[16] Z. Wen, R. Zhang, K. Ramamohanarao, J. Qi, and K. Taylor, "Mascot: Fast and highly scalable svm cross-validation using GPUs and SSDs," in *Proc. IEEE Int. Conf. Data Mining*, 2014, pp. 580–589.

[17] D. Lee *et al.*, "On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 1999, pp. 134–143.

[18] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2014, pp. 2040–2048.

[19] M. Z. Shafiq, A. X. Liu, and A. R. Khakpour, "Revisiting caching in content delivery networks," in *Proc. ACM Int. Conf. Meas. Model. Comput. Syst.*, 2014, pp. 567–568.

[20] I. Psaras, W. K. Chai, and G. Pavlou, "In-network cache management and resource allocation for information-centric networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2920–2931, Nov. 2014.

[21] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 95–107, Jan. 2016.

[22] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3665–3677, Oct. 2014.

[23] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 4, pp. 2995–3007, Apr. 2016.

[24] J. Gu, W. Wang, A. Huang, and H. Shan, "Proactive storage at caching-enable base stations in cellular networks," in *Proc. IEEE Annu. Int. Symp. Personal, Indoor, Mobile Radio Commun.*, 2013, pp. 1543–1547.

[25] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: Why it matters and how to model it," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 5–12, Nov. 2013.

[26] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1863–1876, Aug. 2016.

[27] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. Annu. Conf. Inf. Sci. Syst.*, 2018, pp. 1–6.

[28] S. Lee, I. Yeom, and D. Kim, "T-caching: Enhancing feasibility of in-network caching in ICN," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1486–1498, Jul. 2020.

[29] A. Ioannou and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2847–2886, Oct.–Dec. 2016.

[30] J. Yang, C. Ma, J. Man, H. Xu, G. Zheng, and H. Song, "Cache-enabled in cooperative cognitive radio networks for transmission performance," *Tsinghua Sci. Technol.*, vol. 25, no. 1, pp. 1–11, 2020.

[31] G. Li *et al.*, "Data-driven approaches to edge caching," in *Proc. Workshop Netw. Emerg. Appl. Technol.*, 2018, pp. 8–14.

[32] Z. Zhao, M. Peng, Z. Ding, W. Wang, and H. V. Poor, "Cluster content caching: An energy-efficient approach to improve quality of service in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1207–1221, May 2016.

[33] W. Sun, Y. Li, C. Hu, and M. Peng, "Joint optimization of cache placement and bandwidth allocation in heterogeneous networks," *IEEE Access*, vol. 6, pp. 37250–37260, 2018.

[34] N. Wang, G. Shen, S. K. Bose, and W. Shao, "Zone-based cooperative content caching and delivery for radio access network with mobile edge computing," *IEEE Access*, vol. 7, pp. 4031–4044, 2019.

[35] T. Nie, J. Luo, L. Gao, F.-C. Zheng, and L. Yu, "Cooperative edge caching in small cell networks with heterogeneous channel qualities," in *Proc. IEEE Veh. Technol. Conf.*, 2020, pp. 1–6.
[36] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 281–294, Feb. 2020.
[37] J. Choi, A. S. Reaz, and B. Mukherjee, "A survey of user behavior in VoD service and bandwidth-saving multicast streaming schemes," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 1, pp. 156–169, Jan.–Mar. 2012.
[38] G. Yu and J. Wu, "Content caching based on mobility prediction and joint user prefetch in mobile edge networks," *Peer-to-Peer Netw. Appl.*, vol. 13, no. 5, pp. 1839–1852, 2020.
[39] T. X. Tran, D. V. Le, G. Yue, and D. Pompili, "Cooperative hierarchical caching and request scheduling in a cloud radio access network," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2729–2743, Dec. 2018.
[40] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.
[41] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 47–54.
[42] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[43] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," 2017, *arXiv:1704.02971*.
[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[45] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM'99 Conf. Comput. Commun.*, 1999, pp. 126–134.
[46] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.
[47] P. Blasco and D. G'nd'z, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int. Conf. Commun.*, 2014, pp. 1897–1903.
[48] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. Uncertainty Artif. Intell.*, 2020, pp. 367–377.
[49] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2015.

Xu Zhang received the BS degree in communication engineering from the Beijing University of Posts and Telecommunications, China, in 2012 and the PhD degree in computer science from the Department of Computer Science and Technology, Tsinghua University, China, in 2017. He is currently a Marie Sklodowska-Curie research fellow with the College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, U.K. His research interests include artificial intelligence, multimedia communication, and network measurement. He was the co-recipient of 2019 IEEE Broadcast Technology Society Best Paper Award.

Zhengnan Qi received the BE degree in communication engineering from the Ji luan Academy Nanchang University, Nanchang, China, in 2019, and is currently working toward the master's degree with the School of Electronic Science and Engineering, Nanjing University, Naning, China. His research interests include content delivery networks, content caching, and multimedia communications.

Geyong Min received the BSc degree in computer science from the Huazhong University of Science and Technology, China, in 1995, and the PhD degree in computing science from the University of Glasgow, U.K., in 2003. He is currently a professor of high-performance computing and networking with the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences, University of Exeter, U.K. His research interests include computer networks, wireless communications, parallel and distributed computing, ubiquitous computing, multimedia systems, modelling, and performance engineering.

Wang Miao received the PhD degree in computer science from the University of Exeter, U.K., in 2017. He is currently a postdoctoral research associate with the Department of Computer Science, University of Exeter, U.K. His research interests include network function virtualization, software-defined networking, unmanned aerial networks, wireless communication networks, wireless sensor networks, and edge artificial intelligence.

Qilin Fan received the BS degree from the College of Software Engineering, Sichuan University, Chengdu, China, in 2011, and the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2017. She is currently a lecturer with the School of Big Data and Software Engineering, Chongqing University, Chongqing, China. Her research interests include content delivery networks, network measurement, and multimedia communications.

Zhan Ma received the BS and MS degrees from the Huazhong University of Science and Technology, Wuhan, China, in 2004 and 2006, respectively, and the PhD degree from the New York University, New York, NY, USA, in 2011. He is currently the faculty with the Electronic Science and Engineering School, Nanjing University, Jiangsu, China. From 2011 to 2014, he was with Samsung Research America, Dallas TX, and Futurewei Technologies, Inc., Santa Clara, CA, USA. His research interests focuses on the neural video coding and smart cameras. He was the co-recipient of 2018 ACM SIGCOMM Student Research Competition Finalist, 2018 PCM Best Paper Finalist, and 2019 IEEE Broadcast Technology Society Paper Award.