ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20190303092125
Original scientific paper

Cooperative Caching with Content Popularity Prediction for Mobile Edge Caching

Sanshan SUN, Wei JIANG, Gang FENG, Shuang QIN, Ye YUAN

Abstract: Mobile Edge Caching (MEC) can be exploited for reducing redundant data transmissions and improving content delivery performance in mobile networks. However, under the MEC architecture, dynamic user preference is challenging the delivery efficiency due to the imperfect match between users' demands and cached content. In this paper, we propose a learning-based cooperative content caching policy to predict the content popularity and cache the desired content proactively. We formulate the optimal cooperative content caching problem as a 0-1 integer programming for minimizing the average downloading latency. After using an artificial neural network to learn content popularity, we use a greedy algorithm for its approximate solution. Numerical results validate that the proposed policy can significantly increase content cache hit rate and reduce content delivery latency when compared with popular caching strategies.

Keywords: caching; cooperative; mobile edge caching; neural network

1 INTRODUCTION

In the vision of next generation mobile system, content-rich multimedia and cloud application (e.g. C2X-automotive, massive IoT, and e-health, etc.) lead to tremendously increasing data traffic and pose a challenge to the network capacity and performance [1]. A study [2] revealed that a major portion of the increased mobile traffic is the duplicate downloads of a few popular content items from remote servers. Using mobile content caching and delivery techniques, popular content items can be cached in the intermediate servers (such as gateways, routers) so that requests for the same content can be available without duplicate transmissions from remote servers [3], such that significantly improve user Quality of Experience (QoE) while saving transmission resource consumption at the backhaul and core networks.

Content distribution networks (CDN) have been well investigated in the context of wired networks [4], such as the Internet. However, we cannot simply apply traditional CDN-based content distribution technique to mobile networks. It is difficult to meet the dynamic needs of the users in mobile networks, as legacy CDN−based content distribution mechanism is generally designed for traditional wired communication network architecture. In mobile networks, the resources (e.g., storage, bandwidth, and computing capacity) and the position of the deployed servers are constrained. More importantly, the hit rate of cached contents could be rather low in mobile networks due to the content dynamics, user mobility and the limited number of users in a cell. Besides, the amount of content provided by Content Providers (CPs) is growing rapidly and it is thus impossible to cache all contents, although storage cost is becoming much cheaper. Therefore, it is imperative to design efficient content caching strategies to maximize the benefits of local content caching for future mobile networks.

The emerging Mobile Edge Caching (MEC) technology is currently being standardized by the European Telecommunications Standards Institute (ESTI) Industry Specification Group (ISG) [5], which can be exploited to accelerate the development of intelligent content caching at mobile network edges. MEC architecture provides a high possibility to cache popular content items in close proximity to mobile subscribers [6, 7]. Under this circumstance, we can acquire accurate cell information (such as users' demands, radio conditions, etc.) dynamically for performing optimal content caching. Moreover, MEC servers can perform caching in a cooperative way and share their cached content items with each other [8]. Intuitively caching popular content items at multiple MEC servers in a cooperative way can further improve local cache hit rate and thus alleviate user perceived latency.

There are two types of mobile caching strategies: those with perfect content popularity information and those without. Content popularity is the probability that a specific content item is requested at a certain time. In the first type of strategies, the author of [9] assumes the popularity of content items is known or obeys Zipf distribution or its variants. The authors of [10] investigated the limits of caching in a caching system with a single server and multiple users. The authors of [11] proposed coordinated caching at base-station (BS) to reduce redundant transmissions over the core networks. The authors of [12] studied a caching problem in a heterogeneous network with helpers, in which users can connect to several helpers, and fetch files from the helper that offers the least latency. The problem was shown to be NP-hard, and a heuristic was proposed. In [13], wireless bandwidth constraints are set in the design of a caching policy of small BSs. The authors of [14] modeled the Device-to-Device (D2D) content caching distribution as a Zipf distribution similar to the content demand distribution to improve the utilization of network resources. In [15], the authors propose an optimal cooperative caching policy for HetNets with femtocells and D2D communications, with the aim of providing a global optimal caching solution for HetNets. These studies assumed the perfect knowledge of instantaneous content demands or the content popularity matrix, which is rather unrealistic.

In the second type of strategies, MEC servers can be used to track and predict users' demand, and *proactively* cache popular content items at mobile network edges [16]. In [17] and [18], the authors proposed a transfer learning-based caching procedure to exploit the rich contextual information extracted from the source domain. This prior information is incorporated in the target domain for finding the best contents to cache at a small cell. The caching problem in small BS is modeled as a combinatorial multiarmed bandit (MAB) problem in [19]. In [20], an upper confidence bounds (UCB)-type algorithm is used to find the content popularity matrix. The matrix is then used to optimize the cache content placement, taking into account users' connectivity to the small BS. These studies considered either a single cache case or multiple cache cases without cooperation. For multiple cache cases with cooperative content caching and delivery, a new policy is needed. Moreover, the successful application of some intelligent learning algorithms in data correlation analysis [21, 22] and probability distribution estimation [23] also inspires us to use similar schemes to predict content mobility, thereby improving the content caching efficiency under the MEC architecture.

In this paper, we propose a learning-based cooperative caching framework for connected MEC servers that needs no prior knowledge of content demands and content popularity matrix. We first use an artificial neural network to learn the content popularity matrix by observing the instantaneous demands over time. We then formulate content caching for minimum latency problem as a 0-1 integer-programming problem. We further show that the problem is NP-hard and propose a greedy algorithm for its solution.

We present the system model and formulate the problem in Section 2. In Section 3, we present the neural network and the greedy algorithm. Finally, we evaluate the performance of our proposed caching policy by simulation in Section 4 and conclude the paper in Section 5.

2 SYSTEM MODEL AND PROBLEM FORMULATION

2.1 System Model and Assumptions

The purpose of MEC is to reduce latency and network traffic congestion by delegating computation and content caching from the core WAN to the edge network. Fig. 1 shows our system model.

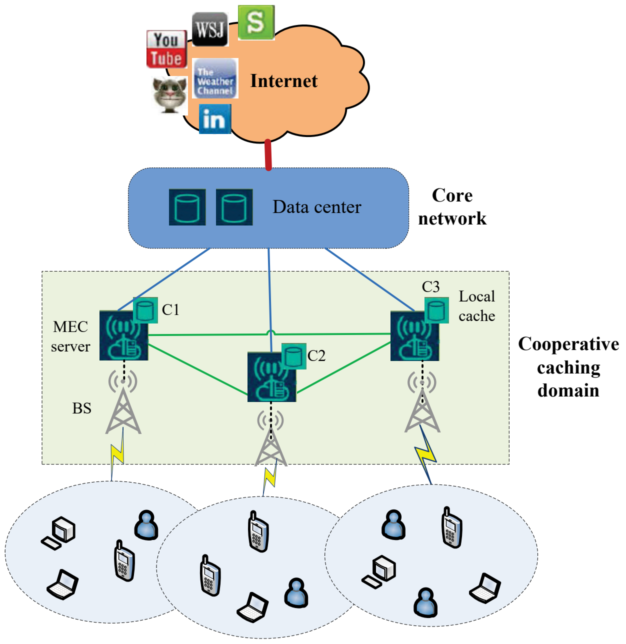


Figure 1 A cooperative caching domain based on MEC

It is shown that content providers are offering content over the Internet via data centers. Data centers deploy popular content items in local MEC servers located at BSs to reduce redundant transmissions. A collection of MEC servers in a region forms a cooperative caching domain. One MEC server is in charge of content caching and delivery for users in a specific cell [6].

In a specific domain, MEC servers work cooperatively share content items [8]. Thus, a content item missing in C1 may be fetched from C2 or C3 via, for example, the X2 interface between NodeBs in LTE [24], instead of fetching from the data center.

In this paper, we focus on cooperative content caching strategy in a specific cooperative caching domain. Let $\mathcal{C}=\{c_1,c_2,\cdots,c_M\}$ be the set of MEC servers with corresponding storage capacity $\mathcal{S}=\{S_1,S_2,\cdots,S_M\}$ in that domain. Users can make requests from a finite content library set $\mathcal{O}=\{o_1,o_2,\cdots,o_N\}$ of N content items provided by CPs, where each content item o_k is of size s_k.

In the system, time is slotted into intervals and caching solutions are periodically updated for each interval. Let $\tilde{p}_m^k$ denote the popularity of content item o_k in c_m, i.e., the ratio of the number of content requests for o_k in c_m to the total number of content requests in c_m. In practice, the content popularity may change over time and cannot be known in advance. Thus, it must be learned over time.

Consider the following three cases when a content request for content item o_k in c_m needs to be handled: 1) If o_k has already been cached in c_m, then o_k is delivered from c_m to the user directly; 2) If o_k is not cached in c_m but has been distributed to at least one other MEC server in the cooperative caching domain, then c_m fetches content item o_k from other MEC servers; 3) If o_k is not cached in any MEC server in the cooperative caching domain, then c_m fetches content item o_k from the data center. Let $\tau_{m,n}^k$ denote the downloading latency for c_m to fetch content item o_k from c_n. Particularly, let $\tau_{m,0}^k$ denote the downloading latency for c_m to fetch content item o_k from the remote data center via the backhaul link and $\tau_{m,0}^k > \tau_{m,n}^k$ since downloading from data center needs more hops [25].

2.2 Problem Formulation

Based on the discussions above, the cooperative content caching problem addressed in this paper can be described as follows: for a given network topology, MEC servers' storage capacity and random content requests from users, how should the content items be deployed in the local caches of MEC servers such that the average downloading latency is minimized ?

We first define a binary caching decision matrix $\boldsymbol{X}=\{x_m^k \mid c_m \in \mathcal{C}, o_k \in \mathcal{O}\}$, where binary caching decision variable $x_m^k \in \{0,1\}$ indicates whether the content item o_k is placed at the cache of MEC server c_m or not: $x_m^k = 1$ if o_k is cached in c_m and $x_m^k = 0$ otherwise. Thus, the problem of optimal cooperative content caching which maximizes the sum reduction in downloading latency, or equivalently, minimizes the sum downloading latency, can be formulated as,

$$\max_{\mathbf{X}} \sum_{c_m \in \mathcal{C}} \sum_{o_k \in \mathcal{O}} \tilde{p}_m^k \cdot \Big[\tau_{m,0}^k - (1 - x_m^k) \cdot \cdot \sum_{i=1}^{|\mathcal{C}|-1} \Big(\tau_{m,(i)_m}^k \cdot \prod_{j=1}^{i-1} (1 - x_{(j)_m}^k) \cdot x_{(i)_m}^k \Big) - \tau_{m,0}^k \cdot \prod_{n=1}^{M} (1 - x_n^k) \Big], \tag{1}$$

$$s.t. \;\; \textstyle\sum_{o_k \in \mathcal{O}} x_m^k \cdot s_k \le S_m, \forall c_m \in \mathcal{C}, \tag{1.1}$$

$$x_m^k \in \{0,1\}, \forall c_m \in \mathcal{C}, \forall o_k \in \mathcal{O}, \tag{1.2}$$

where $c_{(i)_m}$ is the MEC server with the ith lowest latency for c_m and $\sum_{i=1}^{|\mathcal{C}|-1} (\tau_{m,(i)_m}^k \cdot \prod_{j=1}^{i-1} (1 - x_{(j)_m}^k) \cdot x_{(i)_m}^k)$ is the lowest latency for c_m to fetch content o_k from other MEC servers. Inequality (1.1) indicates the storage capacity constraint.

In problem (1), we need to derive the caching decision matrix $\boldsymbol{X}$. To this end, we need to know the popularity of individual content items first. Next, we will exploit machine learning to predict content popularity based on historical user content demands.

3 LEARNING-BASED COOPERATIVE CONTENT CACHING POLICY

3.1 Content Popularity Prediction by Using Artificial Neural Network

Artificial neural networks are able to learn from examples and capture subtle functional relationships among the data even if the underlying relationships are unknown or hard to describe [26]. This property makes artificial neural networks useful for solving problems whose solutions require knowledge that is difficult to specify but for which there are enough data or observations. In the content caching system, content popularity changes dynamically, while instantaneous content demand can be observed. This inspires us to use an artificial neural network to predict content popularity based on historical demands information.

The artificial neural network model is shown in Fig. 2. The parallel artificial neural network model has M parallel neurons. Each neuron has an input vector $\boldsymbol{p}(t-1) = [1, p_1(t-1), p_2(t-1), ..., p_M(t-1)]^{\mathrm{T}}$ and a connection weight vector $\boldsymbol{w}_i = [w_{i0}, w_{i1}, w_{i2}, \ldots, w_{iM}]^{\mathrm{T}}$. The output of each neuron is the inner product of the input vector and connection weight vector $h_i(t) = \sum_{j=1}^{M} w_{ij} \cdot p_j(t-1) + w_{i0} = \boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{p}(t-1)$.

We define $p_i^k(t) = d_i^k(t) \big/ \sum_{o_j \in \mathcal{O}} d_i^j(t)$ to be the popularity of content item o_k in c_i until time t, where $d_i^k(t)$ is the number of requests for o_k in c_i until time t. For a given input vector

$$\boldsymbol{p}^k(t-1) = [1, p_1^k(t-1), p_2^k(t-1), ..., p_M^k(t-1)]^{\mathrm{T}}, \tag{2}$$

where $p_i^k(t-1)$ is the popularity of content item o_k in c_i until time t−1, we can obtain the output $h_i^k(t) = \boldsymbol{w}_i^T \boldsymbol{p}^k(t-1)$, which is the predicted popularity of content item o_k in c_i until time t. In order to make the predicted content popularity approximate to the ground truth content popularity, we need to learn the optimal connection weight matrix $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_M]$ by training the artificial neural network based on historical content popularity.

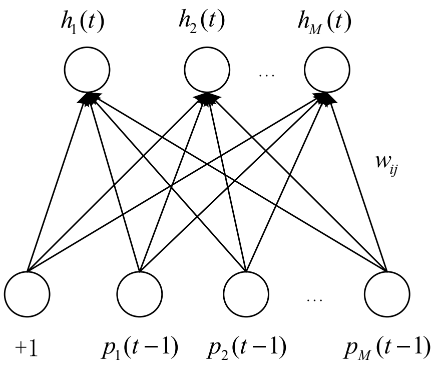


Figure 2 Parallel artificial neural network model

For a given sample $(\boldsymbol{p}^k(t-1), p_i^k(t))$, the error function is defined as:

$$E_i^k \left[\boldsymbol{w}_i \middle| \boldsymbol{p}^k(t-1), p_i^k(t) \right] = 1/2 \left[p_i^k(t) - h_i^k(t) \right]^2 = = 1/2 \left[p_i^k(t) - \boldsymbol{w}_i^{\mathrm{T}} \boldsymbol{p}^k(t-1) \right]^2 \tag{3}$$

Hence, for the training set

$$\boldsymbol{\chi} = \{ \boldsymbol{p}^k(t-1), p_i^k(t) | \forall c_i \in \mathcal{C}, \forall o_k \in \mathcal{O} \}$$

with $M \times N$ samples, the global error function is:

$$E[\boldsymbol{W} | \boldsymbol{\chi}] = \frac{1}{M \times N} \sum_{c_i \in \mathcal{C}} \sum_{o_k \in \mathcal{O}} E_i^k \left[\boldsymbol{w}_i \middle| \boldsymbol{p}^k(t-1), p_i^k(t) \right] \tag{4}$$

In order to derive the optimal $\boldsymbol{W}$ to minimize the global error function value, we take the derivative of the global error function with respect to w_{ij}(i = 1, 2,…, M, j = 0, 1,…, M). Then we can obtain the normal equations:

$$\begin{aligned}
&\sum_{o_k \in \mathcal{O}} p_i^k(t) = N w_{i0} + w_{i1} \sum_{o_k \in \mathcal{O}} p_1^k(t-1) + w_{i2} \sum_{o_k \in \mathcal{O}} p_2^k(t-1) + \\
&+ \cdots + w_{iM} \sum_{o_k \in \mathcal{O}} p_M^k(t-1), \\
&\sum_{o_k \in \mathcal{O}} p_1^k(t-1) p_i^k(t) = w_{i0} \sum_{o_k \in \mathcal{O}} p_1^k(t-1) + w_{i1} \sum_{o_k \in \mathcal{O}} (p_1^k(t-1))^2 + \\
&+ w_{i2} \sum_{o_k \in \mathcal{O}} p_1^k(t-1) p_2^k(t-1) + \cdots + w_{iM} \sum_{o_k \in \mathcal{O}} p_1^k(t-1) p_M^k(t-1), \\
&\sum_{o_k \in \mathcal{O}} p_2^k(t-1) p_i^k(t) = w_{i0} \sum_{o_k \in \mathcal{O}} p_2^k(t-1) + w_{i1} \sum_{o_k \in \mathcal{O}} p_1^k(t-1) p_2^k(t-1) + \\
&+ w_{i2} \sum_{o_k \in \mathcal{O}} (p_2^k(t-1))^2 + \cdots + w_{iM} \sum_{o_k \in \mathcal{O}} p_2^k(t-1) p_M^k(t-1), \\
&\cdots \\
&\sum_{o_k \in \mathcal{O}} p_M^k(t-1) p_i^k(t) = w_{i0} \sum_{o_k \in \mathcal{O}} p_M^k(t-1) + w_{i1} \sum_{o_k \in \mathcal{O}} p_1^k(t-1) p_M^k(t-1) + \\
&+ w_{i2} \sum_{o_k \in \mathcal{O}} p_2^k(t-1) p_M^k(t-1) + \cdots + w_{iM} \sum_{o_k \in \mathcal{O}} (p_M^k(t-1))^2.
\end{aligned} \tag{5}$$

We define matrices $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$ as follows:

$$\mathbf{P}=\begin{bmatrix}1 & p_1^1(t-1) & p_2^1(t-1) & \cdots & p_M^1(t-1)\\ 1 & p_1^2(t-1) & p_2^2(t-1) & \cdots & p_M^2(t-1)\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & p_1^N(t-1) & p_2^N(t-1) & \cdots & p_M^N(t-1)\end{bmatrix}_{N\times(M+1)} \tag{6}$$

$$\hat{\mathbf{P}}=\begin{bmatrix}p_1^1(t) & p_2^1(t) & \cdots & p_M^1(t)\\ p_1^2(t) & p_2^2(t) & \cdots & p_M^2(t)\\ \vdots & \vdots & \ddots & \vdots\\ p_1^N(t) & p_2^N(t) & \cdots & p_M^N(t)\end{bmatrix}_{N\times M} \tag{7}$$

Hence, the normal equations can be rewritten as:

$$\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{W}=\boldsymbol{P}^{\mathrm{T}}\hat{\boldsymbol{P}} \tag{8}$$

Now we can find the optimal connection weight for minimizing the global error function value as

$$\boldsymbol{W}=(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P})^{-1}\boldsymbol{P}^{\mathrm{T}}\hat{\boldsymbol{P}} \tag{9}$$

After the optimal connection weight matrix is obtained, for the time period [t, t+1), the input vector of the artificial neural network is the content popularity until time t. We can obtain the predicted popularity $h_i^k(t+1), \forall c_i\in\mathcal{C}, \forall o_k\in\mathcal{O}$ for the time period [t, t+1), by computing the inner product of the input vector and connection weight vector as

$$h_i^k(t+1)=\sum\nolimits_{j=1}^{M} w_{ij}\cdot p_j^k(t)+w_{i0}=\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{p}^k(t) \tag{10}$$

For clarity, we summarize the content popularity prediction algorithm by using an artificial neural network in Algorithm 1.

Algorithm 1 Content popularity prediction

```
For time period $[t, t+1)$
1) Collect the historical content demands.
    i)   Collect the number of content requests for $o_k$ in $c_i$
         until time $t$, denoted as $d_i^k(t), \forall c_i\in\mathcal{C}, \forall o_k\in\mathcal{O}$.
    ii)  Calculate the popularity of content item $o_k$ in $c_i$
         until time $t$, denoted as
         $p_i^k(t)=d_i^k(t)\big/\sum_{o_j\in\mathcal{O}} d_i^j(t), \forall c_i\in\mathcal{C}, \forall o_k\in\mathcal{O}$.
2) Train the artificial neural network.
    i)   Define the input vector as Eq. (2).
    ii)  Define the connection weight matrix as
         $\boldsymbol{W}=[\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M]$, where
         $\boldsymbol{w}_i=[w_{i0}, w_{i1}, w_{i2}, \ldots, w_{iM}]^{\mathrm{T}}, i=1,2,\ldots,M$.
    iii) Define the output as
         $h_i^k(t)=\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{p}^k(t-1), \forall c_i\in\mathcal{C}, \forall o_k\in\mathcal{O}$.
    iv)  Define the training set as
         $\boldsymbol{\chi}=\{\boldsymbol{p}^k(t-1), p_i^k(t) \mid \forall c_i\in\mathcal{C}, \forall o_k\in\mathcal{O}\}$.
    v)   Define the global error function as Eq. (4).
    vi)  Compute the optimal connection weight $\boldsymbol{W}$ to
         minimize the global error function value.
         a) Take the derivative of the global error
            function with respect to
            $w_{ij}(i=1,2,\ldots,M, j=0,1,\ldots,M)$.
         b) Obtain the normal equations as Eq. (5).
         c) Define matrices $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$ as Eq. (6) and
            Eq. (7) respectively.
         d) Rewrite the normal equations as Eq. (8).
         e) Compute the optimal connection weight for
            minimizing the global error function value
            according to Eq. (9).
3) Compute the predicted content popularity.
    i)   Let the input vector of the artificial neural network
         be the content popularity until time $t$:
         $\boldsymbol{p}^k(t), \forall o_k\in\mathcal{O}$.
    ii)  Compute the output of the artificial neural
         network as Eq. (10).
    iii) Update the content popularity of $o_k$ in $c_i$ as
         $\tilde{p}_i^k=h_i^k(t+1)$.
```

3.2 Sub-optimal Solution for Cooperative Content Caching

We can solve the cooperative content caching problem (1) with the predicted content popularity. We first prove Lemma 1.

Lemma 1. The cooperative content problem (1) is NP-hard even with the knowledge of content popularity $\tilde{p}_m^k$ at each MEC server.

Proof: Consider a special case of the problem where $\tau_{m,n}^k=0$ and $s_k=s$ for all $c_m, c_n\in\mathcal{C}$ and $o_k\in\mathcal{O}$. In this case, problem (1) becomes

$$\max_{\boldsymbol{X}} \sum\nolimits_{c_m\in\mathcal{C}}\sum\nolimits_{o_k\in\mathcal{O}} p_m^k\cdot\tau_{m,0}^k\cdot\left[1-\prod\nolimits_{n=1}^{M}(1-x_n^k)\right] \tag{11}$$

$$s.t.\ \sum\nolimits_{o_k\in\mathcal{O}} x_m^k\le S_m/s, \forall c_m\in\mathcal{C} \tag{11.1}$$

$$x_m^k\in\{0,1\}, \forall c_m\in\mathcal{C}, \forall o_k\in\mathcal{O} \tag{11.2}$$

It is a generalization of the Helper Decision Problem (HDP), which has been proven to be NP-hard [13], and thus problem (1) is also NP-hard.

As we cannot solve the optimal cooperative caching problem in polynomial time, we resort to a greedy algorithm based on the predicted content popularity targeting at a sub-optimal solution. Our main ideas are as follows. We first propose a metric, namely Maximum Unit Value (MaxUV), defined as the maximum unit value obtained by placing a content item to a non-full cache. We start with an empty cache set of MEC servers and then place the content items in an iterative way. In each iteration, we compute the current objective value of problem (1) by caching o_k in c_m, denoted as Z_m^k, and the maximum unit value can be defined as $\text{MaxUV}=\max\limits_{\forall c_m\in\mathcal{C}, \forall o_k\in\mathcal{O}} Z_m^k/s_k$. Then o_k is cached in c_m according to the MaxUV. The algorithm terminates when all the caches become full. We can summarize the greedy algorithm in Algorithm 2.

In the greedy algorithm for the cooperative content caching problem, there would be at most $\sum_{c_m\in\mathcal{C}} S_m$ iterations until all the caches are filled. Each iteration takes $O(NM)$ time. Hence the computational complicity for Algorithm 2 is $O(NM\cdot\sum_{c_m\in\mathcal{C}} S_m)$.

Algorithm 2 Greedy algorithm for the cooperative content caching problem

```
1) Initialize content caching decision matrix $X = 0$.
2) for $\sum_{o_k \in \mathcal{O}} x_m^k \cdot s_k < S_m, \exists c_m \in \mathcal{C}$ do
3)   for each $c_m \in \mathcal{C}$ and $o_k \in \mathcal{O}$ do
4)     if content $o_k$ has not been placed in $c_m$ and
       $\sum_{o_i \in \mathcal{O}} x_m^i \cdot s_i + s_k \le S_m$ then
5)       Compute the objective value of problem (1) with the
         assumption of $x_m^k = 1$ based on the current $X$, defined as
         $Z_m^k$.
6)     end if
7)   end for
8)   Find the maximum value of $Z_m^k / s_k, \forall c_m \in \mathcal{C}, \forall o_k \in \mathcal{O}$, and
     update $\mathrm{MaxUV} = \max_{\forall c_m \in \mathcal{C}, \forall o_k \in \mathcal{O}} Z_m^k / s_k$.
9)   Cache content item $o_k$ in $c_m$ according to the MaxUV,
     and update $x_m^k = 1$, $Z_m^k = 0$.
10) end for
11) Obtain the content caching decision matrix $X$.
```

4 NUMERICAL RESULTS

In this section, we evaluate the performance of our proposed learning-based cooperative content caching policy. The performance metrics we employ include the average delivery latency (ADL) and cache hit rate (CHR). ADL is defined as the ratio of the sum delivery latency for all content requests to the total number of content requests. CHR is defined as $HR \triangleq H/R$, where R is the total number of content requests, and H is the number of content requests which are served by a MEC server in the cooperative caching domain. Obviously, the higher cache hit rate, the better content delivery performance. For computing the delivery latency for a request for o_k in c_m, we need to consider three cases: 1) If c_m has cached o_k, the latency is 0. 2) If c_m has not cached o_k and c_n is the closet MEC server of c_m who has cached o_k, and then c_m fetches o_k from c_n. Thus the latency is $\tau_{m,n}^k$. 3) If o_k is not cached in any MEC server in the cooperative caching domain, and then c_m fetches content item o_k from the data center. The latency is hence $\tau_{m,0}^k$.

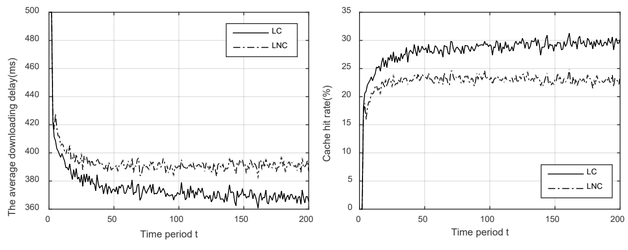
We conduct simulation experiments to compare the performance of our learning-based cooperative content caching (LC) with the following four content caching schemes. 1) Informed Upper Bound (IUB) [19] which assumes that the content popularity matrix is perfectly known in advance and then content items are deployed into MEC servers by using the greedy algorithm proposed in this paper. 2) Learning based Non-Cooperative caching (LNC), in which each MEC server caches content items in a non-cooperative way, i.e., each MEC server caches the most popular content items based on the predicted content popularity which is learned by using the artificial neural network proposed in this paper; 3) Least Recently Used (LRU) [26] which removes the least recently used content item and caches the content item that is requested; 4) Random Caching (RC) [27], which randomly caches content items in the MEC servers until all caches become full. Note that no learning happens in the random caching scheme, while the LRU caching scheme learns only from one-step past.

We consider a cooperative caching domain with $M = 4$ MEC servers, each of which is located in a cell. We assume that the content library has N = 2000 items [12] and the items are randomly chosen to have sizes $s_k \in \{1,3,5,7,9\}, \forall o_k \in \mathcal{O}$ units [19]. The cache size of MEC servers is $S_m = 256, \forall c_m \in \mathcal{C}$ units [19]. We assume that content requests in each cell occur independently following a Poisson process with an average rate $\lambda = 50$ (arrivals/time period) [12]. We study 200 time periods and generate a total number of 10^4 content requests for each cell. The popularity distribution of content items is generally modeled as a Zipf distribution [13, 14], i.e., the probability that a request is for the jth most popular content item is $j^{-\theta} / \sum_{i=1}^{N} i^{-\theta}$ and the shape factor θ of ZipF distribution is 0.56 [13]. Since the content popularity in each cell may be different, we assume that 50% of the content items have the same popularity order in all cells and the rest of content items have random popularity order in each cell. The downloading rate $R_{m,n}$ between c_m and c_n is 10 (units/s), and $\tau_{m,n}^k = s_k / R_{m,n}, \forall c_m, c_n \in \mathcal{C}, m \ne n$, $\forall o_k \in \mathcal{O}$. We introduce a parameter $\gamma = \tau_{m,0}^k / \tau_{m,n}^k$ as the ratio of the downloading latency for c_m to fetch o_k from data center to the downloading latency for c_m to fetch o_k from c_n, and assume $\gamma = 5, \forall c_m, c_n \in \mathcal{C}, m \ne n$, in terms of hop count [25]. In summary, the system parameters used in the performance evaluation are listed in Tab. 1.

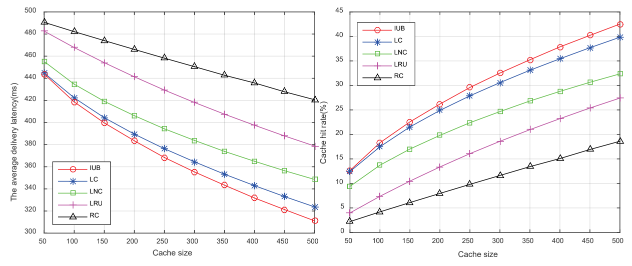
Table 1 System parameters

Number of MEC servers (M)	4
Number of content items (N)	2000
Size of each content item (s_k)	{1,3,5,7,9}
Cache size of MEC servers (S_m)	256
Average arrival rate of content requests in each cell (λ)	50
Total content requests of each cell	10^4
Shape factor of Zipf distribution (θ)	0.56
Downloading rate between each MEC server ($R_{m,n}$)	10
The ratio of the downloading latency (γ)	5

In the first experiment, we examine the time evolution of LC and LNC schemes, i.e., the impact of the number of historical content request samples which are used for learning on the caching performance. Fig. 3 shows the ADL and CHR vs. time period. We can see that the ADL decreases with t, while the CHR increases with t. This is because the more historical content request we collect with t, the more accurate content popularity prediction is. Furthermore, it is shown that both LC and LNC schemes converge quickly and achieve near-optimal performance after 50 time periods.



(a) Average downloading latency (b) Cache hit rate
Figure 3 Performance comparison of LC and LNC for different time periods.

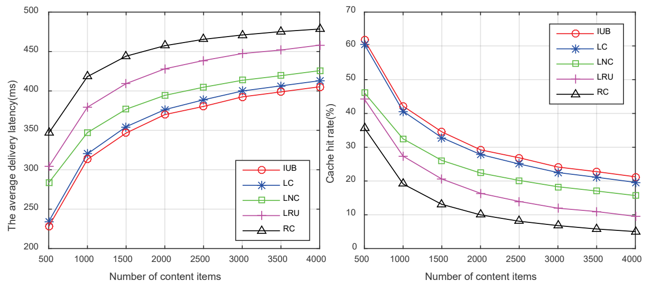


(a) Average downloading latency (b) Cache hit rate

Figure 4 Performance comparison of IUB, LC, LNC, LRU, and RC for different cache sizes of MEC servers.

Next, we compare the ADL and CHR when the storage capacity of MEC servers varies from 50 to 500 units in Fig. 4. As expected, with the increase of the cache size, the ADL decreases and the CHR increases for all caching schemes since more content items can be cached in MEC servers. Note that IUB has the best performance in ADL and CHR, as the content popularity is assumed perfectly known in advance, while RC has the worst performance. LNC outperforms LRU since the learning is used to predict the content popularity. LC outperforms LNC due to the cooperation among MEC servers, and the performance gains of LC are close to that of IUB. In particular, compared with LNC, LRU and RC schemes, the gain of LC in terms of ADL is around 7%, 14%, and 23%, respectively when the cache size is up to 500.

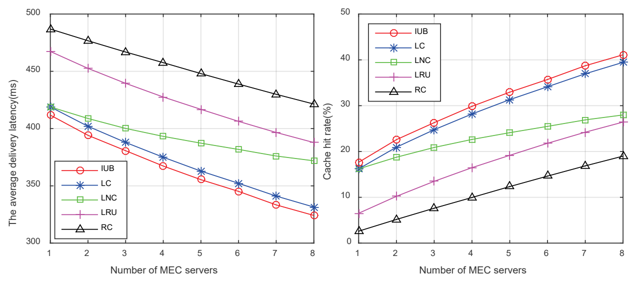
Fig. 5 shows the ADL and CHR as a function of the total number of content items. We can see that the ADL increases with *N*, while the CHR decreases with *N* for all schemes, due to the limited cache size of MEC servers. The performance gain of LC is close to that of IUB due to learning gain. Furthermore, it is shown that LC outperforms the LNC, LRU, and RC with the improvement on CHR up to 25%, 105% and 289%, respectively when the number of content items is up to 4000.



(a) Average downloading latency (b) Cache hit rate

Figure 5 Performance comparison of IUB, LC, LNC, LRU, and RC for different numbers of content items.

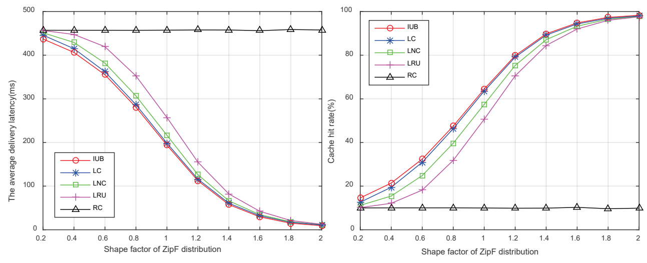
In the following, we compare the ADL and CHR for the different number of MEC servers. In Fig. 6, we observe that the ADL decreases and the CHR increases with the number of MEC servers for all caching schemes. This is because with the increase of the number of MEC servers, more content requests can be satisfied by MEC servers in the cooperative caching domain. Note that the performance difference between LC and LNC becomes more significant when the number of MEC servers increases. Hence, the cooperative caching policy can greatly reduce the average latency and improve the cache hit rate, especially for the case of a large number of MEC servers in a cooperative caching domain. In Fig. 6 (a), the ADL of LC caching policy is significantly lower than that of LNC, LRU and RC: the improvement is approximately 0%-11%, 10%-14% and 14%-21% when compared with LNC, LRU, and RC, respectively for the number of MEC servers ranging from 1 to 8.



(a) Average downloading latency (b) Cache hit rate

Figure 6 Performance comparison of IUB, LC, LNC, LRU, and RC for different numbers of MEC servers.

Finally, we examine the impact of the shape factor of content popularity distribution (Zipf) θ on ADL and CHR in Fig. 7. We can see that when θ ranges from 0.2 to 2, the ADL decreases and the CHR increases for IUB, LC, LNC and LRU schemes. From the figure, we can see that with the increase of θ, the performance gap between the IUB, LC, LNC, and LRU schemes becomes smaller and eventually diminishes. The reasons are as follows. When θ is large, the vast majority of user requests are concentrated on a small number of content items. Clearly, caching the most popular content items provides more significant benefits.



(a) Average downloading latency (b) Cache hit rate

Figure 7 Performance comparison of IUB, LC, LNC, LRU, and RC for different shape factors of ZipF distribution

5 CONCLUSIONS

In this paper, we have proposed a learning-based cooperative content caching scheme for MEC architecture, when the content popularity is unknown and only the historical content demands can be observed. We have formulated the cooperative content caching problem as a 0-1 integer programming aiming to minimize the average downloading latency, and shown that it is NP-hard. The content popularity is learned via an artificial neural network and a greedy algorithm has been proposed to solve the optimization problem based on the predicted popularity. Numerical results show that our proposed learning based cooperative content caching policy outperforms existing known content caching schemes in terms of cache hit rate and average delivery latency.

Acknowledgements

This work is supported in part by the National Science Foundation of China (No. 61471089), Scientific Research Fund of Sichuan Provincial Education Department (No.

17ZB0355), and Open Research Fund of Chongqing E-commerce and Modern Logistics Laboratory (No. ECML01710).

6 REFERENCES

[1] Cisco, (2012). *Cisco visual networking index: Forecast and methodology*, 2011-2016. CISCO White paper, 518.

[2] Erman, J., Gerber, A., Hajiaghayi, M., Pei, D., Sen, S., & Spatscheck, O. (2011). To cache or not to cache: The 3G case. *IEEE Internet Computing, 15*(2), 27-34. https://doi.org/10.1109/MIC.2010.154

[3] Wang, X., Chen, M., Taleb, T., Ksentini, A., & Leung, V. C. (2014). Cache in the air: Exploiting content caching and delivery techniques for 5G systems. *IEEE Communications Magazine, 52*(2), 131-139. https://doi.org/10.1109/MCOM.2014.6736753

[4] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). Mobile edge computing-A key technology towards 5G. *ETSI white paper, 11*(11), 1-16.

[5] Patel, M., Naughton, B., Chan, C., Sprecher, N., Abeta, S., & Neal, A. (2014). Mobile-edge computing introductory technical white paper. *White Paper, Mobile-edge Computing (MEC) industry initiative*, 1089-7801.

[6] Ahmed, A. & Ahmed, E. (2016). A Survey on Mobile Edge Computing. *10th IEEE International Conference on Intelligent Systems and Control, (ISCO 2016)*. https://doi.org/10.1109/ISCO.2016.7727082

[7] Beck, M. T., Werner, M., Feld, S., & Schimper, S. (2014, November). Mobile edge computing: A taxonomy. In *Proc. of the Sixth International Conference on Advances in Future Internet*, Citeseer, 48-55.

[8] Bastug, E., Bennis, M., & Debbah, M. (2014). Living on the edge: The role of proactive caching in 5G wireless networks. *IEEE Communications Magazine, 52*(8), 82-89. https://doi.org/10.1109/MCOM.2014.6871674

[9] Bi, S., Zhang, R., Ding, Z., & Cui, S. (2015). Wireless communications in the era of big data. *IEEE Communications Magazine, 53*(10), 190-199. https://doi.org/10.1109/MCOM.2015.7295483

[10] Maddah-Ali, M. A., & Niesen, U. (2014). Fundamental limits of caching. *IEEE Transactions on Information Theory, 60*(5), 2856-2867. https://doi.org/10.1109/TIT.2014.2306938

[11] Li, Y., Xu, Y., Lin, T., Wang, X., & Ci, S. (2013). A novel coordinated edge caching with request filtration in radio access network. *The Scientific World Journal*, 2013. https://doi.org/10.1155/2013/654536

[12] Shanmugam, K., Golrezaei, N., Dimakis, A. G., Molisch, A. F., & Caire, G. (2011). Femtocaching: Wireless video content delivery through distributed caching helpers. *arXiv preprint arXiv*: 1109.4179.

[13] Poularakis, K., Iosifidis, G., & Tassiulas, L. (2014). Approximation algorithms for mobile data caching in small cell networks. I*EEE Transactions on Communications, 62*(10), 3665-3677. https://doi.org/10.1109/TCOMM.2014.2351796

[14] Malak, D. & Al-Shalash, M. (2014, December). Optimal caching for device-to-device content distribution in 5G networks. In *2014 IEEE Globecom Workshops (GC Wkshps)*, 863-868.
Jiang, W., Feng, G., & Qin, S. (2017). Optimal cooperative content caching and delivery policy for heterogeneous cellular networks. *IEEE Transactions on Mobile Computing, 16*(5), 1382-1393. https://doi.org/10.1109/TMC.2016.2597851

[15] Baştuğ, E., Bennis, M., & Debbah, M. (2016). Proactive caching in 5G small cell networks. In *Towards 5G: Applications, Requirements and Candidate Technologies*, Wiley, 78-98.

[16] Baştuğ, E., Bennis, M., & Debbah, M. (2015, May). A transfer learning approach for cache-enabled wireless networks. In *13th IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 161-166.

[17] Blasco, P. & Gündüz, D. (2014, June). Learning-based optimization of cache content in a small cell base station. In *IEEE International Conference on Communications (ICC)*, 1897-1903. https://doi.org/10.1109/ICC.2014.6883600

[18] Sengupta, A., Amuru, S., Tandon, R., Buehrer, R. M., & Clancy, T. C. (2014, August). Learning distributed caching strategies in small cell networks. In *ISWCS*, 917-921. https://doi.org/10.1109/ISWCS.2014.6933484

[19] Shen, J., Suo, S., Quan, H., Zhao, X., Hu, H., & Jiang, Y. (2008). *3GPP long term evolution: principle and system design*. Posts & Telecom Press, Beijing.

[20] Zhang, D., Jin, D., Gong, Y., Chen, S., & Wang, C. (2015). Research of alarm correlations based on static defect detection. *Tehnicki Vjesnik-Technical Gazette, 22*(2), 311-318. https://doi.org/10.17559/TV-20150317102804

[21] Zhang, D., Sui, J., & Gong, Y. (2017). Large scale software test data generation based on collective constraint and weighted combination method. *Tehnicki Vjesnik-Technical Gazette, 24*(4), 1041-1050. https://doi.org/10.17559/TV-20170319045945

[22] Zhang, W., Zhang, Z., Chao, H. C., & Tseng, F. H. (2018). Kernel mixture model for probability density estimation in Bayesian classifiers. *Data Mining and Knowledge Discovery, 32*(3), 675-707. https://doi.org/10.1007/s10618-018-0550-5

[23] Li, Y., Xie, H., Wen, Y., & Zhang, Z. L. (2013, July). Coordinating in-network caching in content-centric networks: Model and analysis. In *IEEE 33rd International Conference on Distributed Computing Systems*, 62-72.

[24] Cobb, J. & ElAarag, H. (2008). Web proxy cache replacement scheme based on back-propagation neural network. *Journal of Systems and Software, 81*(9), 1539-1558. https://doi.org/10.1016/j.jss.2007.10.024

[25] Abrams, M., Standridge, C. R., Abdulla, G., Williams, S., & Fox, E. A. (1995). *Caching proxies: Limitations and potentials*. Department of Computer Science, Virginia Polytechnic Institute & State University.

[26] Psounis, K., & Prabhakar, B. (2001). A randomized web-cache replacement scheme. In *Proceedings IEEE INFOCOM 2001, Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society* (Cat. No. 01CH37213), Vol. 3, 1407-1415. https://doi.org/10.1109/INFCOM.2001.916636

Contact information:

Sanshan SUN, (Corresponding author)
E-mail: sanshansun@hotmail.com
Wei JIANG,
E-mail: 781806601@qq.com
Gang FENG, Prof. Dr.
E-mail: fenggang@uestc.edu.cn
Shuang QIN, Associate Professor. Dr.
E-mail: blueqs@uestc.edu.cn
National Key Laboratory of Science and Technology on Communications,
University of Electronic Science and Technology of China,
Chengdu, China

Ye YUAN, Associate Professor. Dr.
E-mail: yuanye@cqupt.edu.cn
School of Economics and Management,
Chongqing University of Posts and Telecommunications,
Chongqing, China