A Novel Caching Policy with Content Popularity Prediction and User Preference Learning in Fog-RAN

Yanxiang Jiang[†*], Miaoli Ma[†], Mehdi Bennis[‡], Fuchun Zheng[†§], and Xiaohu You[†]
[†]National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China.
[‡]Centre for Wireless Communications, University of Oulu, Oulu 90014, Finland.
[§]Department of Electronic Engineering, University of York, York YO10 5DD, U.K.
[*]E-mail: yxjiang@seu.edu.cn

***Abstract*—In this paper, the edge caching problem in fog radio access networks (F-RAN) is investigated. By maximizing the cache hit rate, we formulate the edge caching optimization problem to find the optimal edge caching policy. Considering that users prefer to request the contents they are interested in, we propose to implement online content popularity prediction by leveraging the content features and user preferences, and offline user preference learning by using the "Follow The (Proximally) Regularized Leader" (FTRL-Proximal) algorithm and the "Online Gradient Descent" (OGD) method. Our proposed edge caching policy not only can promptly predict the future content popularity in an online fashion with low computational complexity, but also can track the popularity changes in time without delay. Simulation results show that the cache hit rate of our proposed policy approaches the optimal performance and is superior to those of the traditional policies.**

***Index Terms*—F-RAN, edge caching, cache hit rate, content popularity, user preference.**

I. Introduction

With the continuous and rapid proliferation of various intelligent devices and advanced mobile application services, wireless networks have been suffering an unprecedented data traffic pressure in recent years. Ever-increasing mobile data traffic brings tremendous pressure on capacity-limited backhaul links, especially at peak traffic moments. Fog radio access networks (F-RAN) can effectively reduce the backhaul load by placing popular contents as close as possible to the requesting users, and have now attracted more and more attention from researchers and engineers [1]–[3]. In F-RAN, fog access points (F-APs) are equipped with limited caching and computing resources. Due to storage constraints and fluctuant spatio-temporal traffic demands, however, F-APs face a myriad of challenges. For example, how, what and when to strategically store contents in their local caches in order to achieve a higher cache hit rate.

Traditional caching policies such as first in first out (FIFO) [4], least recently used (LRU) [5], least frequently used (LFU) [6] and their variants [7] have been widely applied in wired networks, where there are abundant caching and computing resources and the served area is usually very large. However, traditional caching policies are no longer applicable in F-RAN due to the characteristics of F-APs such as smaller coverage areas and limited caching resources, and they may also suffer major performance degradation since they tend not to predict future content popularity. As a result, recent works have turned to exploring sophisticated edge caching policies by learning future content popularity. In [8], the content popularity matrix was estimated through transfer learning by leveraging user-content correlation and information transfer between time slots. Nevertheless, the content popularity matrix remains typically to be large, which needs a large amount of calculation in the estimation. Meanwhile, the transfer learning approach has a poor performance for the case of low information correspondence ratio. In [9], the cache content placement problem was modeled as a contextual multi-arm bandit problem and an online policy was presented to learn the content popularity. This policy learns the content popularity independently across contents whereas ignores the content similarity and impact of user preference on content popularity, thereby resulting in high training complexity and slow learning speed. A low complexity online policy was proposed in [10], where content popularity was learned based on the assumption that the expected popularity of similar contents are similar. It performs well for video caching but may be ineffective for other types of content caching.

Motivated by the aforementioned discussions, we propose a novel edge caching policy by learning user preference. Our proposed policy can predict the future content popularity in an online fashion and track the popularity change based on the current user preference model and the features of the requested content. It does not need the knowledge of the distribution of content popularity in advance, and does not need continuous offline training, either. On the contrary, it can predict the content popularity directly when the request arrives, and simultaneously offers a higher prediction precision and a lower computational complexity by monitoring the average logistic loss in real time and learning user preference with self-starting in an offline fashion. Specifically, our proposed policy is effective in various content caching scenarios.

The rest of this paper is organized as follows. In section II, the system model and the cache optimization problem are presented. the proposed edge caching policy is described in

Section III. Simulation results are shown in Section IV. Final conclusions are drawn in Section V.

II. System Model

We consider the edge caching problem in a specific region served by M F-APs, which constitute an F-AP set $\mathcal{M} = \{1, 2, \cdots, m, \cdots, M\}$. It is assumed that the users in the specific region can by served by the M F-APs to fetch the contents of interest from their local caches. Without loss of generality, we assume that all the contents have the same size and each F-AP has the same storage space and can store up to φ contents from the content library $\mathcal{F} = \{1, 2, \cdots, f, \cdots, F\}$ located in the cloud content center. By considering the user mobility, the F-APs set the corresponding monitoring cycle according to the characteristics of the specific region that they serve in, and monitor the users in the specific region during discrete time periods $t = 1, 2, \cdots, T$, where T is set to be a finite time horizon. Correspondingly, caching decisions can be made during each time period. Let U_t denote the number of the users served by the M F-APs during the period t, and $\mathcal{U}_t = \{1, 2, \cdots, U_t\}$ the user set monitored by the M F-APs. It is assumed that the users in the region remain unchanged during the considered time period. In the way of cache content placement, for description convenience, we adopt a simple partition-based caching method [11], where each content is separated into M equal-sized partitions, and the F-APs store its different partitions. We remark here that more sophisticated content placement methods can be adopted.

Let D_t denote the number of the requests during the period t, $D = \sum_{t=1}^{T} D_t$ the total number of requests in the finite time horizon T, and $req_t = \{req_{t,1}, req_{t,2}, \cdots, req_{t,d}, \cdots, req_{t,D_t}\}$ the requests which come in sequence during the period t. The request $req_{t,d}$ can be further expressed as: $req_{t,d} = \langle f(d), t(d), \boldsymbol{x}(d) \rangle, \forall 1 \leq t \leq T, \forall 1 \leq d \leq D_t$, where $f(d) \in \mathcal{F}$ denotes the requested content, $t(d)$ the requesting time, $\boldsymbol{x}(d) \in \mathbb{R}^N$ the feature vector of the requested content. $\boldsymbol{x}(d)$ describes the features of the requested content. Take the movie as an example: it may include features like the movie rating, the movie type, the keyword frequency of movie critics, etc. Without loss of generality, we normalize the various dimensions of $\boldsymbol{x}(d)$ and set $\boldsymbol{x}(d) \in [0, 1]^N$.

During the period t, for each arriving request $req_{t,d}$, the F-APs first check whether $f(d)$ has been stored locally. Let $\theta_{t,d}(f(d)) \in \{0, 1\}$ denote the cache status of $f(d)$ for the dth request during the period t, where $\theta_{t,d}(f(d)) = 1$ if $f(d)$ is stored locally, and $\theta_{t,d}(f(d)) = 0$ otherwise. If $f(d)$ has been stored in the local caches, a cache hit happens and the requesting user can then be served locally. Otherwise, $f(d)$ is fetched from the cloud content center, and a caching decision is made to detemine whether to store $f(d)$ in the local caches. If the F-APs decide to store $f(d)$ and replace one of the existing contents in the local caches, denoted by $f_{\text{old}} \in \{f | \theta_{t,d}(f) = 1, \forall f \in \mathcal{F}\}$, a cache update happens. Then, the cache status can be updated according to the following rule,

$$\theta_{t,d+1}(f) = \begin{cases} 0, & f = f_{\text{old}}, \\ 1, & f = f(d), \\ \theta_{t,d}(f), & f \in \mathcal{F} \backslash \{f(d), f_{\text{old}}\}. \end{cases} \tag{1}$$

In addition, a caching decision may be made which determines not to store $f(d)$, and the cache status remains unchanged, i.e., $\theta_{t,d+1}(f) = \theta_{t,d}(f), \forall f \in \mathcal{F}$.

For description convenience, we use $\boldsymbol{\theta}_{t,d} = [\theta_{t,d}(1), \theta_{t,d}(2), \cdots, \theta_{t,d}(f), \cdots, \theta_{t,d}(F)]^T$ to indicate the cache status of all the contents for the dth request during the period t. Generally, an edge caching policy can be represented by a function $\Phi : (\boldsymbol{\theta}_{t,d}, \boldsymbol{x}(d), \mathcal{U}_t) \mapsto \boldsymbol{\theta}_{t,d+1}$, which maps the current cache status vector, the current feature vector, and the current user set to the next cache status vector. After a request $req_{t,d}$ is served, the cache status vector should be updated according to Φ as: $\boldsymbol{\theta}_{t,d+1} = \Phi(\boldsymbol{\theta}_{t,d} | f(d), \boldsymbol{x}(d), \mathcal{U}_t)$. We use cache hit rate to evaluate the caching performance, which is defined as the number of cache hits over the overall requests during the finite time horizon T as follows,

$$A(\Phi) = 1/D \sum\nolimits_{t=1}^{T} \sum\nolimits_{d=1}^{D_t} \theta_{t,d}(f(d)). \tag{2}$$

Then, the corresponding edge caching optimization problem can be expressed as follows,

$$\begin{aligned} &\max_{\Phi} A(\Phi), \\ &\text{s.t. } \theta_{t,d}(f) \in \{0, 1\}, \text{for } 1 \leq d \leq D_t, 1 \leq t \leq T, \forall f \in \mathcal{F}, \\ &\quad \boldsymbol{\theta}_{t,d}^T \boldsymbol{\theta}_{t,d} \leq M\varphi, \text{for } 1 \leq d \leq D_t, 1 \leq t \leq T. \end{aligned} \tag{3}$$

The objective of this paper is to find the optimal edge caching policy by maximizing the cache hit rate over the finite time horizon T with limited total cache size $M\varphi$.

III. The Proposed Edge Caching Policy

In order to maximize the cache hit rate, we propose a novel edge caching policy which includes online content popularity prediction and offline user preference learning. The proposed policy can continuously cache popular contents based on the content features and user preferences.

A. Policy Description

The detailed edge caching policy is shown in Algorithm 1. The considered M F-APs serving in the region set a fixed monitoring period and periodically monitor the current user set in the region. During the period t, the M F-APs first obtain the current user set $\mathcal{U}_t$. For each arriving request $req_{t,d}$ from the user $u \in \mathcal{U}_t$, the access information of the requested user will then be recorded. Meanwhile, the features of $f(d)$ are extracted and recorded. The recorded data will be used to train or update the user preference model in order to improve the prediction precision of the content popularity. Let $\mathcal{G}_{t,d} = \{f | \theta_{t,d}(f) = 1, \forall f \in \mathcal{F}\}$ denote the content set of the current local caches. It will be explored to see whether $f(d)$ has already been stored locally. Just as stated in Section II, the corresponding caching decision will be made to determine whether $f(d)$ should be cached and which stored content

Algorithm 1 The Proposed Edge Caching Policy

procedure EDGECACHING($req_{t,d}$)
  for $t = 1, 2, \cdots, T$, **do**
    The considered M F-APs monitor $\mathcal{U}_t$;
    for $d = 1, 2, \cdots, D_t$, **do**
      Record the access information of $req_{t,d}$;
      Read the current caching content set $\mathcal{G}_{t,d}$;
      if $f(d)$ has been stored locally, **then**
        The users are served locally;
        $P^{\text{cur}}_{t,f(d)} = P^{\text{cur}}_{t,f(d)} - 1/U_t$;
      else
        Fetch $f(d)$ from the cloud content center;
        Extract $\boldsymbol{x}(d)$ and $\{\boldsymbol{w}_u | \forall u \in \mathcal{U}_t\}$;
        $\widehat{P}_{t,d} \leftarrow \text{Predict}(\boldsymbol{x}(d), \{\boldsymbol{w}_u | \forall u \in \mathcal{U}_t\})$;
        Sort $Q_{t,d}$ based on $P^{\text{cur}}_{t,f}$ and t_f for $f \in \mathcal{G}_{t,d}$;
        Get P^{least}, f^{least} from the top of $Q_{t,d}$;
        if $\widehat{P}_{t,d} > P^{\text{least}}$, **then** ▷ Cache update
          Remove the top element from $Q_{t,d}$;
          $t_{f(d)} = t(d)$; $P^{\text{cur}}_{t,f(d)} = \widehat{P}_{t,d} - 1/U_t$;
          Insert $\langle P^{\text{cur}}_{t,f(d)}, f(d), t_{f(d)} \rangle$ into $Q_{t,d}$;
          Replace f^{least} by $f(d)$.
        end if
      end if
    end for
  end for
end procedure

should be removed from the storage space of the M F-APs when $f(d)$ needs to be cached.

In order to make optimal caching decision, the feature vector $\boldsymbol{x}(d)$ and the well-trained user preference model parameters of all the users $\{\boldsymbol{w}_u | \forall u \in \mathcal{U}_t\}$ are extracted to predict the popularity $P_{t,d}$ of $f(d)$. In addition, considering that the content popularity will change over time, in order to track popularity changes, we let $P^{\text{cur}}_{t,f}$ denote the current popularity of the caching content $f \in \mathcal{G}_{t,d}$ after the content is requested (also called the residual request rate). We know that users may have a certain access delay on the same content. In order to ensure timely and reasonable cache updating, we propose to select the content with the characteristics of the least content popularity P^{least} and relatively earlier initial cache time $t_{f^{\text{least}}}$, denoted by f^{least}, as the content to be removed from the local caches. In order to locate f^{least} quickly, we propose to reserve a priority queue $Q_{t,d}$ that stores the caching contents along with their current content popularity $P^{\text{cur}}_{t,f}$ and their initial caching time t_f for $f \in \mathcal{G}_{t,d}$. The elements of $Q_{t,d}$ are sorted in sequence when the request $req_{t,d}$ arrives, whose top element is composed of f^{least}, $t_{f^{\text{least}}}$ and P^{least}. A caching decision is made by comparing the predictive popularity $\widehat{P}_{t,d}$ and P^{least}. If $\widehat{P}_{t,d}$ is larger than P^{least}, the existing content f^{least} will be replaced by $f(d)$, the initial caching time of $f(d)$ will be recorded, and the current popularity of $f(d)$ and $Q_{t,d}$ will be updated accordingly. After that, a cache update process is completed. Otherwise, nothing will be done to the local caches. The key here, obviously, is to obtain $\widehat{P}_{t,d}$, which is described in the next subsection.

B. Online Content Popularity Prediction

In this subsection, we will propose an online content popularity prediction based on the content features and user preferences. During the period t, for each requesting user, the requested content can be classified into a favorite category and unfavorite category for this user based on its user preference. Generally, a user prefers to request contents of its favorite category. The problem of whether a user will request a certain content can be converted into a simple two-category one. We use the sigmoid function to approximate the correspondence between the feature vector and the category label of the requested content [12], and construct a logical regression model to approximate the user preference model. For the arriving request $req_{t,d}$, it is characterized by the feature vector $\boldsymbol{x}(d)$. Let $y(d)$ denote the corresponding category label with $y(d) = 1$ if the requested content is the favorite category of the user and $y(d) = 0$ otherwise. Let $p_{t,u,d}$ denote the possibility that the user $u \in \mathcal{U}_t$ requests the content $f(d)$ at the requesting time $t(d)$ during the period t. Specifically, we assume that a user will not have a second request to the same content. If the user u has already requested $f(d)$ previously, then $\widehat{p}_{t,u,d} = 0$. Otherwise, $p_{t,u,d}$ can be predicted based on the following user preference model,

$$\widehat{p}_{t,u,d} = p_{\boldsymbol{w}_u}(y(d) = 1 | \boldsymbol{x}(d)) = 1/(1 + e^{-(\boldsymbol{w}_u * \boldsymbol{x}(d))}). \quad (4)$$

Furthermore, the content popularity $P_{t,d}$ for the considered region can be predicted by using the average requested possibility of the users in this region to $f(d)$ as follows,

$$\widehat{P}_{t,d} = 1/U_t \sum\nolimits_{u=1}^{U_t} \widehat{p}_{t,u,d}. \quad (5)$$

To measure the prediction performance, we introduce the logistic loss $\ell(\boldsymbol{w}_u, \boldsymbol{x}(d), y(d))$ for the user u, which is defined as the negative log-likelihood of $y(d)$ given $p_{\boldsymbol{w}_u}(y(d) | \boldsymbol{x}(d))$ and can be expressed as follows,

$$\begin{aligned} \ell(\boldsymbol{w}_u, \boldsymbol{x}(d), y(d)) = &-y(d) \log p_{\boldsymbol{w}_u}(y(d) | \boldsymbol{x}(d)) \\ &- (1 - y(d)) \log[1 - p_{\boldsymbol{w}_u}(y(d) | \boldsymbol{x}(d))]. \end{aligned} \quad (6)$$

Since user preference may change over time, we need to capture the moment when the user preference changes and timely update the user preference model. For this purpose, we collect the samples $\{(\boldsymbol{x}^{(k)}, y^{(k)})\}_{k=1}^{K_{t,u,d}}$ and monitor the prediction performance in real time, where $\boldsymbol{x}^{(k)}$ and $y^{(k)}$ denote the feature vector and category label of the kth sample, respectively, and $K_{t,u,d}$ the cumulative number of samples for user u from the last model update to the time when the request $req_{t,d}$ arrives during time period t. Then, the average logistic loss for user u can be expressed as follows,

$$\xi_{t,u,d} = 1/K_{t,u,d} \sum\nolimits_{k=1}^{K_{t,u,d}} \ell(\boldsymbol{w}_u, \boldsymbol{x}^{(k)}, y^{(k)}). \quad (7)$$

Let γ denote a predefined threshold with $0 \leq \gamma \leq 1$. When $\xi_{t,u,d}$ exceeds γ, the user preference model update will be

Algorithm 2 Online Content Popularity Prediction

procedure PREDICT($\boldsymbol{x}(d), \{\boldsymbol{w}_u | \forall u \in \mathcal{U}_t\}$)
  for $u \in \mathcal{U}_t$, **do**
    if User u has requested the content before, **then**
      $\widehat{p}_{t,u,d} = 0$;
    else
      Obtain the well-trained $\boldsymbol{w}_u$ from the M F-APs;
      $\widehat{p}_{t,u,d} = 1/(1 + e^{-(\boldsymbol{w}_u * \boldsymbol{x}(d))})$;
      Observe the category label $y(d)$ of $f(d)$.
    end if
    Get $(\boldsymbol{x}^{(K_{t,u,d})}, y^{(K_{t,u,d})})$; $K_{t,u,d} = K_{t,u,d-1} + 1$;
    $\xi_{t,u,d} = \xi_{t,u,d-1} + \frac{1}{K_{t,u,d}} \ell(\boldsymbol{w}_u, \boldsymbol{x}^{(K_{t,u,d})}, y^{(K_{t,u,d})})$;
    if $\xi_{t,u,d} \geq \gamma$, **then**
      $\boldsymbol{w}_u \leftarrow$ Learn$(\{(\boldsymbol{x}^{(k)}, y^{(k)})\}_{k=1}^{K_{t,u,d}})$.
    end if
  end for
  return $\frac{1}{U_t} \sum_{u=1}^{U_t} \widehat{p}_{t,u,d}$.
end procedure

initiated.

The detailed description of our proposed online content popularity prediction approach is shown in Algorithm 2. We remark here that our proposed approach not only can predict content popularity in an online fashion, but also can determine when to update the user preference model proactively. We also remark that the computational complexity of our proposed approach is very low.

C. Offline User Preference Learning

When the user preference model update is initiated, we assume that there are K samples, denoted by $\left\{\left(\boldsymbol{x}^{(k)}, y^{(k)}\right)\right\}_{k=1}^{K}$, collected for the considered user, which are extracted from the recorded data. Based on the collected samples, we propose to learn the user preference model parameter by minimizing the logistic loss of each sample.

Due to the sparse and unbalanced data and high-dimensional feature vector, there may exist over-fitting and high computational complexity problems [13]. In order to avoid the above mentioned potential problems whereas obtain the optimal model parameter, we propose to introduce the $L1$- and $L2$-regularization terms simultaneously in the optimization problem that minimizes the logistic loss. The introduction of $L1$-regularization term is beneficial for realizing feature selection and producing sparse model, while the introduction of $L2$-regularization term is conductive to the smooth solution of the corresponding optimization problem. Let $\boldsymbol{w}_u^{(k)}$ denote the user preference model parameter for the kth iteration. Then, it can be updated according to the previous sample by solving the following optimization problem,

$$\begin{aligned}\boldsymbol{w}_u^{(k+1)} = \underset{\boldsymbol{w}_u}{\operatorname{argmin}}\{&\ell(\boldsymbol{w}_u, \boldsymbol{x}^{(k)}, y^{(k)}) \\ &+\lambda_1 \|\boldsymbol{w}_u\|_1 + 1/2\lambda_2 \|\boldsymbol{w}_u\|_2^2\},\ k = 1, 2, \cdots, K, \quad (8)\end{aligned}$$

where $\lambda_1, \lambda_2 > 0$ denote the regularization parameters, $\|\cdot\|_1$ and $\|\cdot\|_2^2$ the $L1$-norm and $L2$-norm, respectively.

By using the "Follow The (Proximally) Regularized Leader" (FTRL-Proximal) algorithm [14], which is an online optimization algorithm, combined with the "Online Gradient Descent" (OGD) method [15], the above optimization problem can be transformed into the following equivalent form,

$$\begin{aligned}\boldsymbol{w}_u^{(k+1)} = \underset{\boldsymbol{w}_u}{\operatorname{argmin}}\{&(\boldsymbol{g}^{(1:k)} - \sum\nolimits_{r=1}^{k} \sigma^{(r)} \boldsymbol{w}_u^{(r)})^T \boldsymbol{w}_u \\ &+1/2(\lambda_2 + \sum\nolimits_{r=1}^{k} \sigma^{(r)}) \|\boldsymbol{w}_u\|_2^2 + \lambda_1 \|\boldsymbol{w}_u\|_1 \\ &+1/2 \sum\nolimits_{r=1}^{k} \sigma^{(r)} \|\boldsymbol{w}_u^{(r)}\|_2^2\},\ k = 1, 2, \cdots, K, \quad (9)\end{aligned}$$

where $\boldsymbol{g}^{(1:k)} = \sum_{r=1}^{k} \boldsymbol{g}^{(r)}$ denotes the sum of the gradient vector of the logistic loss of the previous k samples, $\boldsymbol{g}^{(r)} = \nabla_{\boldsymbol{w}_u} \ell\left(\boldsymbol{w}_u, \boldsymbol{x}^{(r)}, y^{(r)}\right) = \left[p_{\boldsymbol{w}_u}\left(y^{(r)} \middle| \boldsymbol{x}^{(r)}\right) - y^{(r)}\right] \boldsymbol{x}^{(r)}$ denotes the gradient vector of the logistic loss of the rth sample with respect to $\boldsymbol{w}_u$, and $\sigma^{(r)}$ satisfies $\sum_{r=1}^{k} \sigma^{(r)} = 1/\eta^{(k)}$, where $\eta^{(k)}$ denotes a non-increasing learning-rate schedule. It can be readily seen from (9) that $1/2 \sum_{r=1}^{k} \sigma^{(r)} \|\boldsymbol{w}_u^{(r)}\|_2^2$ is irrespective with $\boldsymbol{w}_u$. Let

$$\boldsymbol{z}^{(k)} = \boldsymbol{g}^{(1:k)} - \sum\nolimits_{r=1}^{k} \sigma^{(r)} \boldsymbol{w}_u^{(r)}. \quad (10)$$

Then, an iterative relationship between $\boldsymbol{z}^{(k)}$ and $\boldsymbol{z}^{(k-1)}$ can be established as follows,

$$\boldsymbol{z}^{(k)} = \boldsymbol{z}^{(k-1)} + \boldsymbol{g}^{(k)} - \left(1/\eta^{(k)} - 1/\eta^{(k-1)}\right) \boldsymbol{w}_u^{(k)}, \quad (11)$$

which implies that we only need to store $\boldsymbol{z}^{(k-1)}$ after using the last sample for training. Correspondingly, the optimization problem in (9) can be further expressed as follows,

$$\begin{aligned}\boldsymbol{w}_u^{(k+1)} = \underset{\boldsymbol{w}_u}{\operatorname{argmin}}\{&(\boldsymbol{z}^{(k)})^T \boldsymbol{w}_u + \lambda_1 \|\boldsymbol{w}_u\|_1 \\ &+1/2(\lambda_2 + \sum\nolimits_{r=1}^{k} \sigma^{(r)}) \|\boldsymbol{w}_u\|_2^2\},\ k = 1, 2, \cdots, K. \quad (12)\end{aligned}$$

We know that there exists difference for the change rate of the weight of each feature dimension for the requested content, and that the gradient value with respect to each feature dimension can reflect this change rate. Therefore, different learning rates are preferred for different feature dimensions. Define

$$\begin{aligned}\boldsymbol{g}^{(r)} &= [g_1^{(r)}, g_2^{(r)}, \cdots, g_n^{(r)}, \cdots, g_N^{(r)}]^T, \\ \boldsymbol{z}^{(k)} &= [z_1^{(k)}, z_2^{(k)}, \cdots, z_n^{(k)}, \cdots, z_N^{(k)}]^T, \\ \boldsymbol{w}_u &= [w_{u,1}, w_{u,2}, \cdots, w_{u,n}, \cdots, w_{u,N}]^T, \\ \boldsymbol{w}_u^{(k+1)} &= [w_{u,1}^{(k+1)}, w_{u,2}^{(k+1)}, \cdots, w_{u,n}^{(k+1)}, \cdots, w_{u,N}^{(k+1)}]^T.\end{aligned}$$

Let

$$\sum\nolimits_{r=1}^{k} \sigma_n^{(r)} = 1/\eta_n^{(k)},$$

where $\eta_n^{(k)} = \alpha/[\beta + \sqrt{\sum_{r=1}^{k} (g_n^{(r)})^2}]$ denotes the learning-rate schedule of the nth feature dimension with α and β being the adjusting parameters which are chosen to yield good learning performance. Then, the optimization problem in (12) can be decoupled into the following N independent scalar

Algorithm 3 Offline User Preference Learning

procedure LEARN($\{(\boldsymbol{x}^{(k)}, y^{(k)})\}_{k=1}^{K}$)
  Initialize: α, β, λ_1, λ_2, $\boldsymbol{w}_u^{(1)}$, $\boldsymbol{z}^{(0)} = \boldsymbol{q}^{(0)} = \mathbf{0} \in \mathbb{R}^N$;
  for $k = 1, 2, 3, \cdots, K$, **do**
    $\boldsymbol{g}^{(k)} = \nabla \ell_{\boldsymbol{w}_u}(\boldsymbol{w}_u, \boldsymbol{x}^{(k)}, y^{(k)})\big|_{\boldsymbol{w}_u = \boldsymbol{w}_u^{(k)}}$;
    for $n = 1, 2, 3, \cdots, N$, **do**
      $\sigma_n^{(k)} = \frac{1}{\alpha}(\sqrt{q_n^{(k-1)} + (g_n^{(k)})^2} - \sqrt{q_n^{(k-1)}})$;
      $z_n^{(k)} = z_n^{(k-1)} + g_n^{(k)} - \sigma_n^{(k)} w_n^{(k)}$;
      $q_n^{(k)} = q_n^{(k-1)} + (g_n^{(k)})^2$;
      Caculate $w_{u,n}^{(k+1)}$ according to (16) by setting $\sum_{r=1}^{k} \sigma_n^{(r)}$ to $(\beta + \sqrt{q_n^{(k)}})/\alpha$.
    end for
  end for
  return $\boldsymbol{w}_u^{(K+1)}$
end procedure

minimization problems,

$$w_{u,n}^{(k+1)} = \underset{w_{u,n}}{\operatorname{argmin}}\{z_n^{(k)} w_{u,n} + \lambda_1 |w_{u,n}| + 1/2(\lambda_2 + \sum_{r=1}^{k} \sigma_n^{(r)}) w_{u,n}^2\}, \ n = 1, 2, \cdots, N, \quad (13)$$

It can be easily verified that the optimization problem in (13) is an unconstrained non-smooth one, where the second item $\lambda_1 |w_{u,n}|$ is non-differential at $w_{u,n} = 0$. Let $\eta = \partial |w_{u,n}| \Big|_{w_{u,n} = w_{u,n}^{(k+1)}}$ denote the subderivative of $|w_{u,n}|$ at $w_{u,n}^{(k+1)}$. Then, we have

$$\begin{cases} -1 < \eta < 1, & \text{if } w_{u,n}^{(k+1)} = 0, \\ \eta = -1, & \text{else if } w_{u,n}^{(k+1)} < 0, \\ \eta = 1, & \text{otherwise.} \end{cases} \quad (14)$$

From (13), the optimal solution $w_{u,n}^{(k+1)}$ should satisfy the following relationship,

$$z_n^{(k)} + \lambda_1 \eta + (\lambda_2 + \sum_{r=1}^{k} \sigma_n^{(r)}) w_{u,n}^{(k+1)} = 0, \quad n = 1, 2, \cdots, N. \quad (15)$$

We have known previously that $\lambda_1 > 0$. Correspondingly, by classifying $z_n^{(k)}$ into three cases, i.e., $|z_n^{(k)}| < \lambda_1$, $z_n^{(k)} > \lambda_1$ and $z_n^{(k)} < -\lambda_1$, the closed-form solution of the optimization problem in (13) can be obtained from (15) as follows,

$$w_{u,n}^{(k+1)} = \begin{cases} 0, & \text{if } |z_n^{(k)}| < \lambda_1, \\ \frac{\lambda_1 \operatorname{sgn}(z_n^{(k)}) - z_n^{(k)}}{\lambda_2 + \sum_{r=1}^{k} \sigma_n^{(r)}}, & \text{otherwise,} \end{cases} \quad n = 1, 2, \cdots, N. \quad (16)$$

The entire user preference learning approach with the nice property of self-starting is described in Algorithm 3. Note that our proposed approach only needs to store the last $\boldsymbol{z}^{(k)}$ after performing a user preference updating process, and the previously recorded data can be cleared which is helpful to save storage space. Furthermore, the computational complexity of our proposed approach is not an issue due to its offline property.

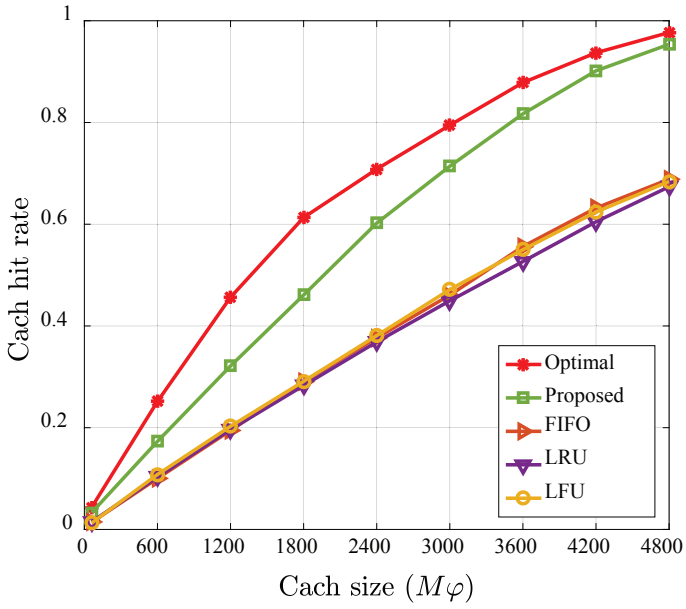

Fig. 1. Cache hit rate versus the total cache size for the proposed policy and the benchmark policies.

IV. Simulation Results

To evaluate the performance of the proposed edge caching policy, we take movie content as an example and our main datasets are extracted from the MovieLens 200M Dataset [16], [17]. From the MovieLens, we select the accessing dataset of the chosen 30 users requested from January 01, 2010 to October 17, 2016, where the first part of the accessing dateset, whose requesting date is from January 01, 2010 to December 31, 2015, is used for initializing the user preference, while the second part of the accessing dateset, whose requesting date is from January 01, 2016 to October 17, 2016, is used for the performance evaluation. Considering that users will generally comment on the movie after they have watched it, we take the movie rating from a user as the request for this movie [9], [10]. In our simulations, we set the number of the considered F-APs M to 3, the finite time horizon T to 6984 hours, the monitoring cycle to 1 hour, and the predefined threshold γ to 0.2, respectively. We choose the FIFO [4], LRU [5], LFU [6] policies, and the optimal policy with ideal content popularity as the benchmark edge caching policies.

In Fig. 1, we show the cache hit rates of our proposed policy and the four benchmark policies with different cache sizes during the finite time horizon T. The total cache size $M\varphi$ increases from $1.5‰F = 60$ to $11.97\%F = 4800$ contents with $F = 40110$. It can be observed that the cache hit rates of all the considered policies are gradually increased with the total cache size. It can also be observed that the cache hit rate of our proposed policy gradually approaches the optimal performance and is apparently superior to those of the other

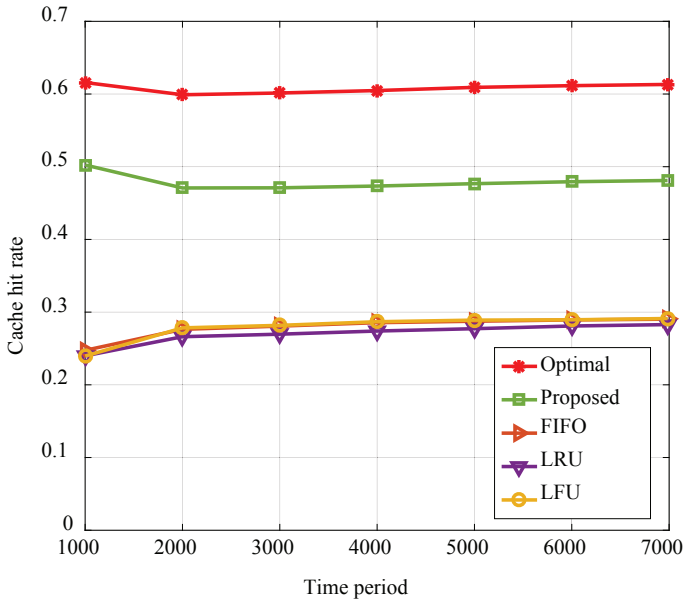

Fig. 2. Cache hit rate versus time period for the proposed policy and the benchmark policies.

three benchmark policies for all the considered cache sizes. The reason is that the latter can not predict future content popularity. Instead, our proposed policy not only can predict the content popularity online, but also can track its changes in real time. Specifically, it can be observed that our proposed policy only needs a cache size of 2400 contents to achieve the cache hit rate of 0.6 whereas the other three benchmark policies need a cache size of about 4200 contents.

In Fig. 2, we show the cache hit rates of our proposed policy and the four benchmark policies till the current time period with the total cache size of 1800 contents. It can be observed that the cache hit rate of our proposed policy follows consistently along with that of the optimal policy. The reasons are that both of them are based on content popularity when making caching decisions, and that the distributions of the predicted popularity and real one are consistent during every time period. It can also be observed that the changes of the cache hit rates of all the policies are different. The FIFO, LRU and LFU policies have low cache hit rates during the initial time periods due to the inevitable cold-start problem, whereas our proposed policy can achieve a higher cache hit rate and cache the predicted popular contents based on the initial user preference during the initial time periods. After that, the cache hit rates of all the policies gradually increase with the time period. The reason is that caching decisions can be made more accurate with the increase of user requests.

V. Conclusions

In this paper, we have proposed a novel edge caching policy by learning user preference. Our proposed policy can promptly detect the regional popular content through online content popularity prediction, and timely store it to the local caches. Specially, we have proposed a self-starting offline user preference model updating mechanism by monitoring the average logistic loss in real time, which avoids frequent and blind training. Simulation results have shown that our proposed policy achieves a better caching performance compared to the traditional policies.

Acknowledgments

This work was supported in part by the Ericsson and SEU Cooperation Project (8504000335), the National 863 Project (2015AA01A709), the National Basic Research Program of China (973 Program 2012CB316004), the Natural Science Foundation of China (61521061), and the UK Engineering and Physical Sciences Research Council under Grant EP/K040685/2.

References

[1] E. Zeydan, E. Bastug, M. Bennis, and et al., "Big data caching for networking: moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, Sept. 2016.

[2] M. Peng, S. Yan, K. Zhang, and et al., "Fog-computing-based radio access networks: Issues and challenges," *IEEE Netwk.*, vol. 30, no. 4, pp. 46–53, Jul. 2016.

[3] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[4] C. Aggarwal, J. Wolf, and P. Yu, "Caching on the world wide web," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 1, pp. 1041–4347, Jan. 1999.

[5] H. Ahlehagh and S. Dey, "Video caching in radio access network: Impact on delay and capacity," in *Proc. 2012 IEEE Wireless Commun. Netwk. Conf.*, Apr. 2012, pp. 2276–2281.

[6] X. Wang, M. Chen, T. Taleb, and et al., "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[7] M. Shafiq, A. Liu, and A. Khakpour, "Revisiting caching in content delivery networks," in *Proc. 2014 ACM Int. Conf. Measurement Modeling Comput. Syst.*, Jun. 2014, pp. 567–568.

[8] E. Bastug, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Proc. 13th Int. Symp. Modeling Opt. Mobile, Ad Hoc, Wireless Netwk.*, Jul. 2015, pp. 161–166.

[9] S. Müller, O. Atan, M. Schaar, and et al., "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.

[10] S. Li, J. Xu, M. Schaar, and et al., "Trend-aware video caching through online learning," *IEEE Trans. Multimedia*, vol. 18, no. 12, pp. 2503 – 2516, Dec. 2016.

[11] Z. Chen, J. Lee, T. Quek, and et al., "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 3401 – 3415, May 2017.

[12] V. Raykar, S. Yu, L. Zhao, and et al., "Supervised learning from multiple experts: Whom to trust when everyone lies a bit," in *Proc. 26th Annu. Int. Conf. Machine Learning*, Jun. 2009, pp. 889–896.

[13] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Datamining, inference, and prediction,*, 2nd ed. New York, NY, USA: Springer, 2009.

[14] H. MacMahan, G. Holt, D. Sculley, and et al., "Ad click prediction: A view from the trenches," in *Proc. ACM SIGKDD 19th Int. Conf. Knowl. Discovery Data Mining*, Apr. 2013, pp. 1222–1230.

[15] P. Bartlett, E. Hazan, and A. Rakhlin, "Adaptive online gradient descent," in *Proc. Neural Inf. Process. Syst. Conf.*, Jan. 2007, pp. 65–72.

[16] F. Harper and J. Konstan, "The movielens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015.

[17] http://files.grouplens.org/datasets/movielens/ml-latest.zip, 2016, [Online; Available 23-Dec-2016].